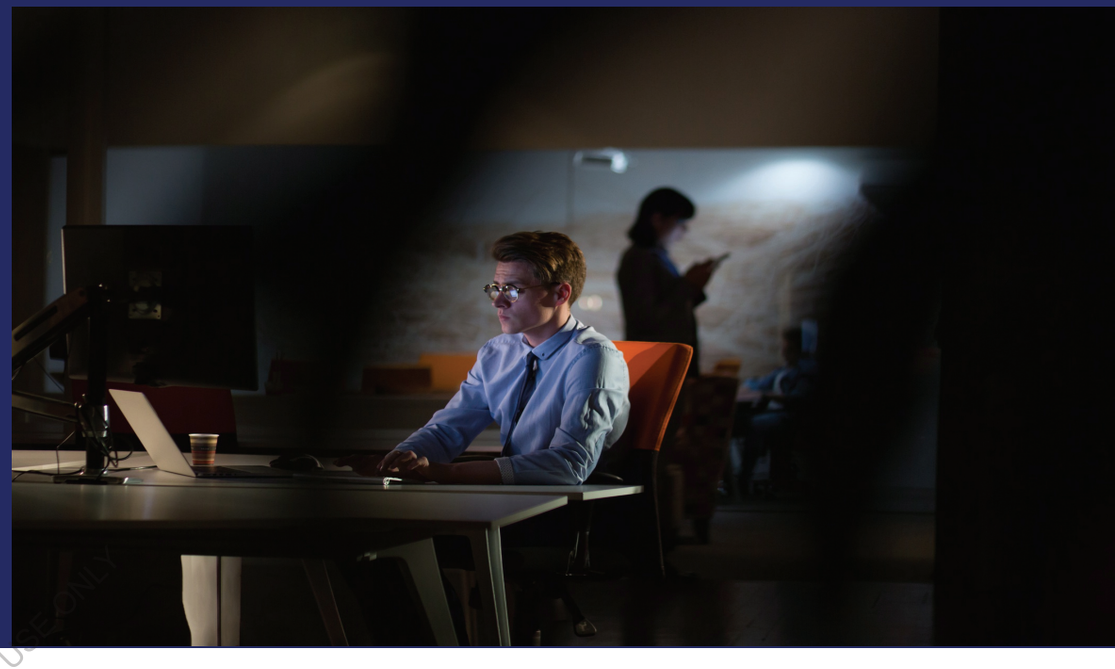
Одной из наиболее важных проблем, являющихся в различных сферах деятельности, является проблема совершенствования управления. Эффективное управление базируется на оптимальном использовании ресурсов и грамотной комплексной оценки организации.
Развитие информационных технологий вызвало появление множества программных продуктов, вызванных автоматизацией деятельности любого предприятия путем полного или частичного устранения человеческого фактора. И если на автоматизированном производстве оборудования вычислима, то в компаниях, где эффективность работы используется человеческих ресурсов, всё не так просто. Если проанализировать вопрос о том, при каком условии можно достичь максимальной производительности труда, то очевидно, что это возможно только при правильном распределении внутренних задач в соответствии с возможностями персонала. Функции адаптации их к специфике предметной области.
Необходимость процесса распределения задач вызвана большим количеством структурных подразделений компаний и отсутствием подходящего программного обеспечения.



Пузырева Д.М. - студентка Набережночелнинского института (филиал) ФГАОУ ВО "Казанского федерального университета"
Еремина И.И. - к.п.н. доцент Набережночелнинского института (филиал) ФГАОУ ВО "Казанского федерального университета"
Лысанов Д.М. - к.т.н. доцент Набережночелнинского института (филиал) ФГАОУ ВО "Казанского федерального университета"

9 786203 928549

Диана Пузырева
Ирина Еремина
Денис Лысанов

Автоматизация распределения задач между сотрудниками

Разработка программного обеспечения для автоматизации распределения задач

LAP LAMBERT Academic Publishing

Диана Пузырева
Ирина Еремина
Денис Лысанов

Автоматизация распределения задач между сотрудниками

Диана Пузырева
Ирина Еремина
Денис Лысанов

Автоматизация распределения задач между сотрудниками

Разработка программного обеспечения для автоматизации распределения задач

LAP LAMBERT Academic Publishing RU

Imprint

Cover image: www.ingimage.com

Publisher:
LAP LAMBERT Academic Publishing
is a trademark of
Dodo Books Indian Ocean Ltd., member of the OmniScriptum S.R.L Publishing group
str. A.Russo 15, of. 61, Chisinau-2068, Republic of Moldova Europe
Printed at: see last page
ISBN: 978-620-3-92854-9

Содержание

Введение

Одной из наиболее важных проблем, возникающих в различных сферах человеческой деятельности, является проблема совершенствования управления. Эффективное управление базируется на оптимальном использовании ресурсов и грамотной комплексной оценке организации.

Развитие информационных технологий вызвало появление множества программных продуктов, призванных автоматизировать деятельность любого предприятия путем полного или частичного устранения человеческого фактора. И если на автоматизированном производстве производительность оборудования вычислима, то в компаниях, где эффективность работы характеризуется использованием человеческих ресурсов, всё не так просто. Если проанализировать вопрос о том, при каком условии можно достигнуть максимальной производительности труда, то очевидно, что это возможно только при правильном распределении внутренних задач в соответствии с возможностями персонала. К особенностям, не позволяющим качественно распределять задачи между сотрудниками в существующих программных продуктах, относится сложность адаптации их к специфике предметной области.

Необходимость автоматизации процесса распределения задач обусловлена большим количеством структурных подразделений компаний и отсутствием подходящего программного обеспечения.

Целью исследовательской работы *является разработка системы управления, которая позволит автоматизировать процесс распределения задач путем эффективного использования трудовых ресурсов.*

Для достижения заданной цели необходимо решить следующие задачи:

1) Проанализировать существующие классификации распределения ресурсов и упорядочения работ предприятия;

2) Изучить критерии оценки эффективности сотрудников;
3) Разработать архитектуру АСУ;
4) Реализовать АСУ;
5) Протестировать АСУ и устранить выявленные недостатки;
6) Внедрить АСУ.

1 Анализ предметной области

1.1 Специфика распределения внутренних задач компании

Для разработки программного обеспечения, предназначенного для автоматизированного распределения задач была исследована работа сотрудников предприятий традиционного организационного устройства и определены особенности, на которые следует уделить особенное внимание в работе:

1. специалисты из смежных отделов способны выполнять задачи друг друга, то есть они являются взаимозаменяемыми в случае необходимости перераспределения подзадач;
2. задачи следует распределять в соответствии с профессиональной квалификацией, что обеспечивается за счет строгого распределения поручений по отделам с учетом индивидуальных достижений каждого сотрудника в предметной области;
3. в любой момент может возникнуть задача, которую необходимо решить в кратчайшие сроки, сместив ранее запланированные дела, но укладываясь в сроки;
4. сроки выполнения задания могут выходить за выделяемые клиентом рамки, о чем необходимо актуально сообщать менеджерам и согласовывать оплату с заказчиками и исполнителями;
5. необходимость систематического контроля исполнительской дисциплины, в простейшем виде рассчитываемой как соотношение количества задач, выполненных работником, к числу задач, которые были поставлены перед ним в расчетном периоде;
6. поставленные перед сотрудником или подразделением задачи могут быть из числа известных и решаемых ранее или новых, направленных на развитие;
7. сотрудники либо являются ориентированными на известные процедуры, либо ориентированы на новые возможности [3].

На основании результатов создана таблица объектов, учитывающих особенности распределения задач и необходимые для корректной работы нового программного продукта. Данные представлены в таблице 1:

Таблица 1. Элементы нового программного продукта

№	Наименование объекта	Использование объекта	Область распространения
1	Исполнительская дисциплина	Объект определяет, какой процент задач выполнялся сотрудником в срок, а какие задачи выполнялись с нарушением заданного клиентом срока.	На конкретного сотрудника
2	Загруженность сотрудника	Объект позволяет определить загрузку сотрудника на заданный период времени, позволяет решить, способен ли он взять на себя новое поручение.	На конкретного сотрудника
3	Сертификация сотрудника	Объект определяет квалификацию сотрудника и имеющиеся у него документы, подтверждающие профессиональный уровень.	На конкретного сотрудника
4	Профессиональная ориентация сотрудника	Объект определяет, является ли сотрудник ориентированным на уже известные процедуры, либо ориентированным на новые возможности.	На конкретного сотрудника

№	Наименование объекта	Использование объекта	Область распространения
5	Процессы с нарушением срока	Объект определяет не выполненные вовремя задачи.	На задачи
6	Задания на согласование	Объект определяет свободные задачи на поручение.	На задачи
7	Динамика заданий на согласование	Объект определяет тенденцию активных задач и завершенных.	На задачи
8	Вид задачи	Объект определяет, какого рода задачи ранее выполнялись и являются задачами функционирования, а с какими ответственный сталкивается впервые, поскольку ранее они решались.	На сотрудника или подразделение
9	Статус	Объект определяет, на каком этапе выполнения находится задача.	На задачу
10	Срок	Объект задает установленный клиентом срок на выполнение.	На задачу
11	Задержка	Объект определяет нарушение срока выполнения задачи.	На задачу

№	Наименование объекта	Использование объекта	Область распространения
12	Отдел назначения	Объект задает род работы для распределения между отделами организации.	На подразделение
13	Ответственный	Объект задает куратора для одного из исполнителей задачи для обучения и контроля качества работы с клиентом. У каждого сотрудника может быть только один куратор, но один и тот же сотрудник может курировать нескольких сотрудников. Также возможна ситуация, когда у сотрудника есть куратор, но и сам этот сотрудник курирует одного или нескольких сотрудников–подопечных.	На конкретного сотрудника

1.2 Постановка задачи

Задача эффективного распределения трудовых задач предприятий традиционного организационного устройства описывается следующим образом:

Основными объектами, необходимыми для автоматизированного распределения заданий на выполнение, являются перечень охарактеризованных задач на согласование, кураторы и исполнители

поручений с установленными критериями эффективности и их загруженность на временной промежуток согласно устанавливаемому сроку.

Существует совокупность задач на выполнение, которые могут быть поставлены перед сотрудником или подразделением. Их можно разделить на два вида: известные, ранее отработанные задачи, они же «задачи функционирования», и ранее не решаемые в компании задачи, «задачи развития». С задачами функционирования лучше справляются специалисты, ориентированные на исполнение задач по отработанным механизмам. Они способны в процессе отработки задачи выявить «узкие места» и поставить в очередь на модернизацию. Благодаря сотрудникам данного типа в организации поддерживается устойчивость и стабильность работы. При этом правильным подходом при обнаружении возможности повышения производительности считается передача задачи для модернизации сотрудникам, ориентированным на решение задач развития. В таком случае задача каждого специалиста заключается в выполнении небольшого объема работ, направленного на достижение общего для компании результата. То есть задача в конкретном случае – это действие или средство, а не сама цель как таковая. Каждая задача определяется и описывается следующим перечнем атрибутов:

- дата начала задачи;
- дата окончания задачи;
- тип задачи;
- исполнитель задачи;
- уровень сложности задачи;
- название задачи;
- предыдущий исполнитель задачи;
- куратор задачи;
- приоритет задачи;
- наиболее ранняя дата начала задачи;

- крайняя дата внедрения задачи;
- описание задачи;
- статус готовности;
- дата сдачи.

Исполнителями являются сотрудники одного из департаментов, потенциально способные взять на себя новое поручение из числа задач или подзадач и выполнить в соответствии с требованиями клиентов. В таком случае сотрудник назначается ожидаемым исполнителем, и задача записывается в число его активных заданий, освобождая от новых поручений до момента окончания текущего. Каждому ожидаемому исполнителю назначается куратор, который формирует состав задач и их техническое описание для более ясного понимания требований заказчиков. Куратор назначается из числа сотрудников компании, но определяется более высокой квалификацией в работе с конкретным поручением для консультирования исполнителя в случае необходимости. Каждый сотрудник описывается следующим перечнем свойств:

- фамилия, имя, отчество исполнителя;
- отдел работы;
- область специализации исполнителя;
- загруженность исполнителя;
- отпуск исполнителя;
- ключевой критерий эффективности исполнителя;
- уровень квалификации исполнителя.

Поскольку куратор и сам является сотрудником, он обладает теми же свойствами, и способен брать на себя задачи, будучи чьим-то подчиненным. Назначение задачи возможно только в случае отсутствия у работника активных поручений, время выполнения которых пересекается со свободным поручением. Загруженность персонала позволяет отследить задания сотрудника по процессам, динамику исполнения заданий (активные задания,

с нарушением срока) и определить, способен ли он взять на себя новую задачу.

Критерии эффективности оценки персонала – совокупность показателей, характеризующих работу каждого сотрудника. Любую деятельность, не вдаваясь в специфику профессии, можно оценить, как со стороны качества выполнения его обязанностей, так и с учетом количества работы и сроков выполнения поручений. Учет задач и функций имеет разную значимость, что порождает разный вес следующих показателей эффективности в системе:

- качество работы (подход к решению задачи, безошибочность, отсутствие жалоб);
- количество работы (объем выполненной работы, норма выработки, доля сверхнормативной выработки);
- срок выполнения работы (соответствие согласованному с клиентом сроку, досрочное выполнение, нарушение сроков).

Итоговая оценка высчитывается как средневзвешенная величина оценок.

При построении плана задач для каждого сотрудника требуется распределить имеющийся перечень задач с учетом ограничений, описанных в разделе Определение ограничений. Необходимо на основе оценки входных задач сформировать список на поручение каждому из сотрудников, в котором оптимизируется функция системы и соблюдаются следующие требования:

- в указанное время каждый сотрудник выполняет только одну задачу;
- время на новое поручение сотруднику, являющемуся куратором, рассчитывается с учетом выделяемого времени на контроль задач подопечных;

- задачи выставлены в промежутке между временем начала и окончания рабочего дня;
- задачи распределены, по возможности, равномерно.

2 Разработка математической модели и алгоритма распределения задач

2.1 Определение основной функции системы, состава системы и ее подсистем

Основная функция системы заключается в формировании списка задач каждого сотрудника.

В состав системы автоматизированного процесса распределения задач входят три основные подсистемы:

1. Подсистема нормативно-справочной информации – данная подсистема должна содержать списки, в которых хранятся данные, используемые при распределении заданий.
2. Подсистема мониторинга загрузки – содержит данные, позволяющие отслеживать занятость сотрудников и получать статистику по их активным и завершенным в настоящий момент задачам.
3. Подсистема планирования – содержит данные для эффективного планирования задачи и загрузки сотрудников в операционной и проектной деятельности в зависимости от нормы рабочего времени.

Результат распределения объектов указан в таблице 2:

Таблица 2. Объекты подсистем

№	Подсистема	Объект подсистемы
1	Подсистема нормативно-справочной информации	Справочник «Структура организации» Справочник «Пользователи» Справочник «Роли» Справочник «Задачи» Справочник «Исполнители» Справочник «Критерии эффективности»

№	Подсистема	Объект подсистемы
		Справочник «Виды отпусков»
2	Подсистема мониторинга загрузки	Справочник «Графики работы сотрудников» Документ «Отпуск сотрудника» Документ «Сертификаты» Объект «Уровни эффективности» Объект «Приоритет задачи» Объект «Уровень сложности» Объект «Типы задач» Объект «Статус готовности сотрудника» Объект «Статусы готовности задачи» Документ «Закрытие задачи» Документ «Изменение статуса задачи» Объект «Отчет о загруженности сотрудников» Объект «Отчет об эффективности сотрудников» Объект «Мои задачи» Объект «Критерии эффективности»
3	Подсистема планирования	Объект «Производственный календарь» Объект «Обмен с мобильным приложением» Объект «Распределение задач» Объект «Назначенные задачи»

2.2 Описание процесса функционирования системы

Процесс автоматизированного распределения задач для пользователя состоит из следующих этапов:

1. вносятся данные нормативно-справочной информации;
2. вносятся данные для мониторинга работы сотрудников;
3. устанавливается норма рабочего времени на определенные календарные периоды времени;
4. рассчитываются ограничения по рабочей нагрузке;
5. пополняется перечень задач на поручение;
6. после внесения необходимой информации, пользователь, используя функционал системы, получает список задач на выполнение, сформированный автоматически;
7. корректируется результат автоматического распределения для получения более эффективного варианта использования трудовых ресурсов.

Процесс алгоритма автоматического распределения задач на основании внесенных данных описан в подразделе 2.6.

Для графического представления описания процессов автоматизации формирования перечня задач были созданы модели в нотации IDEF0 (Рисунок 1, Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5, Рисунок 6):

Рисунок 1 – Схема IDEF0 процесса «Автоматизировать распределение задач» (блок А0)

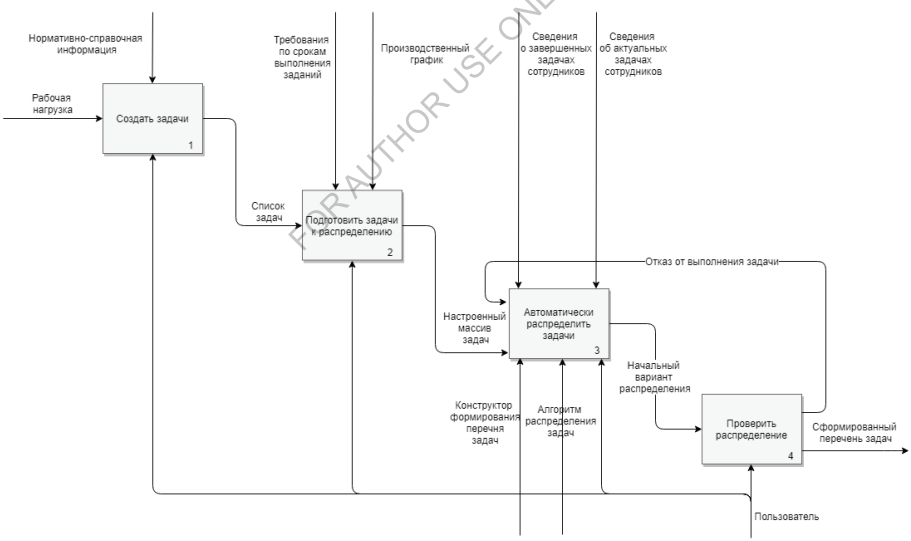


Рисунок 2 – Схема IDEF0 процесса «Автоматизировать распределение задач» (блоки А1-А4)

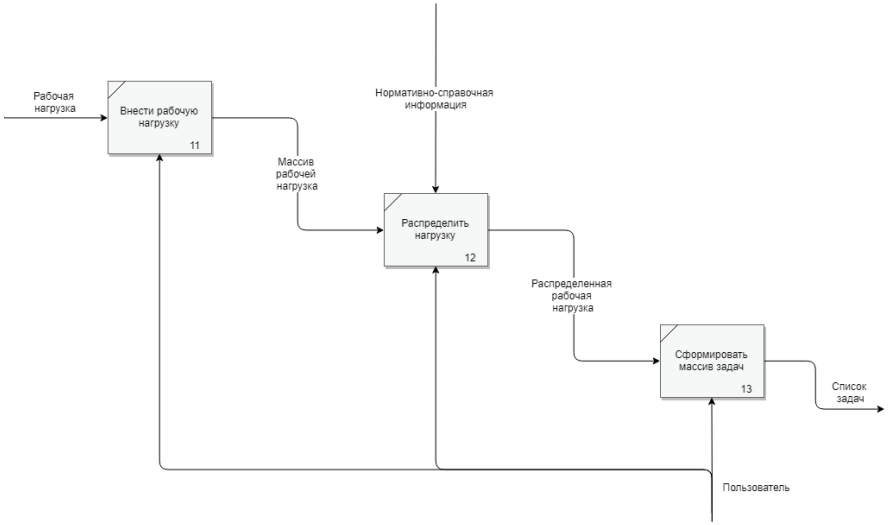


Рисунок 3 – Схема IDEF0 процесса «Создать задачи» (расшифровка блока A1)

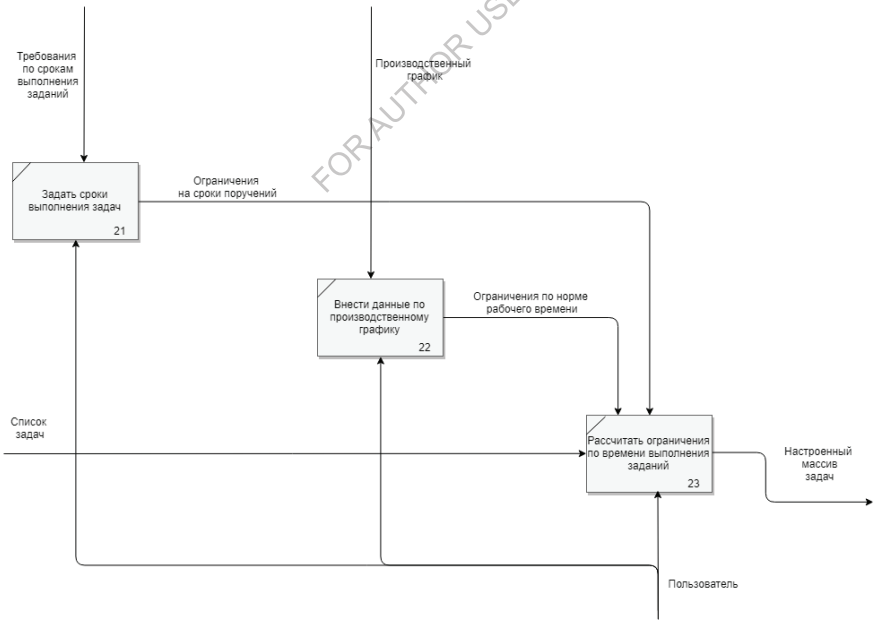


Рисунок 4 – Схема IDEF0 процесса «Подготовить задачи к распределению» (расшифровка блока A2)

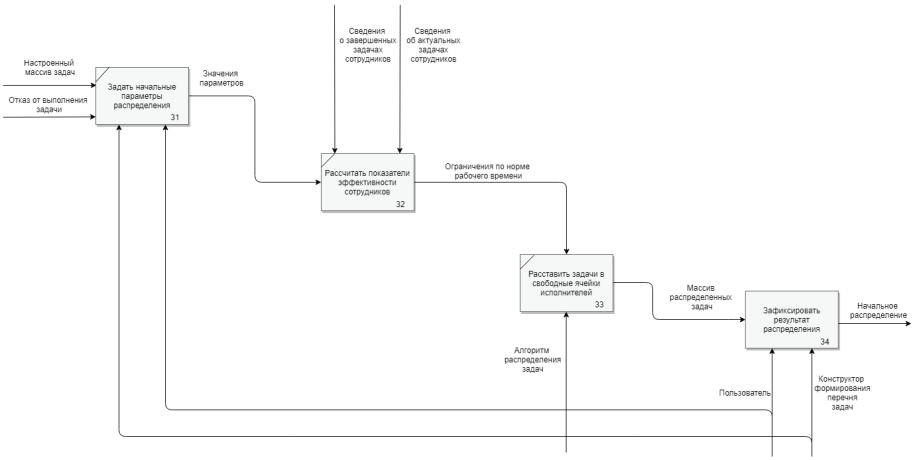


Рисунок 5 – Схема IDEF0 процесса «Автоматически распределить задачи» (расшифровка блока А3)

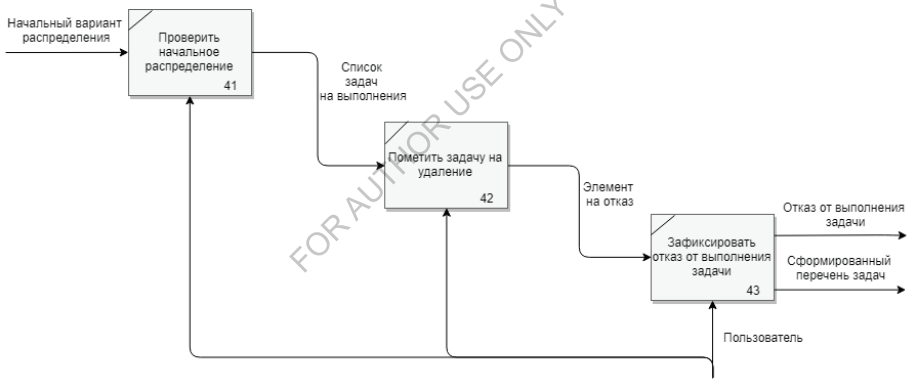


Рисунок 6 – Схема IDEF0 процесса «Проверить распределение» (расшифровка блока А4)

2.3 Определение ограничений

В связи с тем, что необходимые для обеспечения эффективного распределения характеристики трудовых ресурсов, такие как дата начала задачи, дата окончания задачи, приоритет задачи, дата внедрения, отпуск

исполнителя, имеют пределы, то при формировании перечня поручений важно учитывать накладываемые на них ограничения [8][9][10][11].

К данным ограничения относятся:

- Дата начала задачи должна быть больше или равна дате наиболее раннего начала.

$$dt_s(w_i) \geq dt_{min}(w_i)$$

- Дата окончания задачи должна быть меньше либо равна крайней дате внедрения.

$$dt_f(w_i) \leq dt_{max}(w_i)$$

- Дата начала следующей задачи должна быть больше либо равна дате окончания задачи.

$$dt_s(w_{i+1}) \geq dt_f(w_i)$$

- Дата начала задачи не должна быть в период отпуска.

$$dt_s(w_i) \notin \left[dt_s\left(o(e_i)\right);\ dt_f\left(o(e_i)\right)\right]$$

- Дата окончания задачи не должна быть в период отпуска.

$$dt_f(w_i) \notin \left[dt_s\left(o(e_i)\right);\ dt_f\left(o(e_i)\right)\right]$$

- Дата начала задачи должна быть равна дате окончания задачи предшественника.

$$dt_s(w_i) = dt_f(w_{pred(w_i)})$$

- Если приоритет задачи выше приоритета следующей, то дата окончания задачи должна быть меньше даты окончания следующей задачи.

Если $p(w_i) > p(w_{i+1})$, то $dt_f(w_i) < dt_f(w_{i+1})$

2.4 Определение показателей, позволяющих оценить эффективность полученного решения

К показателям, позволяющим оценить эффективность полученного решения, относятся [5]:

- средняя загруженность исполнителей (задачи должны равномерно распределяться между ресурсами);
- индекс сбалансированности загрузки исполнителей (должен стремиться к минимуму);
- сроки проекта (крайняя дата окончания задач должна стремиться к минимуму).

Допустимым списком распределенных задач можно считать список, в котором соблюдаются все ограничения, описанные в подразделе 2.3. Определение ограничений. Эффективным распределением считается такое распределение, в котором значения указанных показателей являются сбалансированными.

2.5 Составление математической модели распределения в общем виде

На основании полученных и проанализированных данных было выполнено составление математической модели процесса распределения задач и трудовых ресурсов, являющегося одной из фундаментальных задач комбинаторной оптимизации, в общем виде.

При распределении задач и трудовых ресурсов используются следующие множества:

- Множество исполнителей $E = \{exec_i, \text{где } i \in [1; N_e]\}$, где N_e - количество исполнителей.
- Множество актуальных задач $W = \{work_j, \text{где } j \in [1; N_w]\}$, где N_w - количество задач.
- Множество завершенных задач $Comp = \{comp_k, \text{где } k \in [1; N_{comp}]\}$, где N_{comp} - количество кураторов.
- Множество кураторов $Cur = \{cur_m, \text{где } m \in [1; N_{cur}]\}$, где N_{cur} - количество кураторов.
- Множество отпусков $V = \{vacat_h, \text{где } h \in [1; N_v]\}$, где N_v - количество отпусков.

Используются следующие матрицы:

- Матрица назначений $P_{e\times w}$, где $p_{ij} = 1$, если утвержден факт i исполнителя на j задачу, иначе $p_{ij} = 0$.
- Матрица ответственных $RESP_{e\times cur}$, где $resp_{im} = 1$, если i сотрудник назначен куратором m исполнителя, иначе $resp_{im} = 0$.
- Матрица критериев эффективности исполнителя $EFF_{e\times comp}$, где eff_{ik} – это действительные показатели характеристик сотрудников на основе завершенных задач.
- Матрица занятости сотрудников $VAC_{e\times v}$, где vac_{ih} – это действительные показатели рабочего графика сотрудников на основе активных задач.

Задача автоматизированного распределения задач и трудовых ресурсов состоит в использовании семейства методов ветвей и графов, основанных на разбиении множества допустимых решений на подмножества, то есть ветвлении, и оценивании целевой функции на этих подмножествах (вычислении границ). В качестве критериев оптимальности полученного решения принимаются показатели, указанные в подразделе Определение показателей, позволяющих оценить эффективность полученного решения [4]. Для удобства и однозначности восприятия критерии эффективности нормируют с учетом следующих правил:

- Средняя загруженность должна стремиться к U_{nom} (задачи должны равномерно распределяться между ресурсами).

$$U = \frac{\sum_{i=1}^{N_w} U_i}{N_e} \rightarrow U_{nom}$$

- Индекс сбалансированности загрузки исполнителей σ. Индекс сбалансированности должен стремиться к минимуму.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N_w}(U_i - U)^2}{N_e - 1}} \rightarrow min$$

- Крайняя дата окончания задач должна стремиться к минимуму.

$$\max(dt_f(w_i)) \rightarrow min$$

Математическая модель данной задачи описывается следующей формулой:

$$\sum_{i=1}^{E}\sum_{j=1}^{W} C(i,m)(\sum_{h}^{V}\sum_{k}^{COMP}\sum_{m}^{CUR} p_{ijhkm}) \rightarrow min$$

При следующих ограничениях:

$\sum_{j=1}^{W} p_{ijhkm} = 1$ для $i \in E$ (1)

$\sum_{i=1}^{E} p_{ijhkm} = 1$ для $j \in W$ (2)

$\sum_{h}^{V} p_{ijhkm} = vac_{ih}$, где

$i = 1, \dots, E, h = 1, \dots, V$ (3)

$\sum_{i=1}^{E} C(i,m)(\sum_{k}^{COMP} p_{ijhkm}) = eff_{ik}$, где

$i = 1, \dots, E, k = 1, \dots, COMP$ (4)

$\sum_{i=1}^{E} C(i,m)(\sum_{m}^{CUR} p_{ijhkm}) \leq resp_{im}$, где

$i = 1, \dots, E, m = 1, \dots, CUR$ (5)

Стоимостная функция $C(i,m)$ определяет стоимость выполнения i исполнителем m задачи.

$C(i,m) = w_1 \sum \tau_i + (1 - w_1) * \left(w_2 \sum c_{ij} + (1 - w_2) \sum {c_{ij}}^{adj}\right)$, где:

- w_1 и w_2 – нормализующие множители;
- $\sum \tau_i$ – сумма вычислительных стоимостей вершин критического пути;

- $c_{ij}{}^{adj}$ – коммуникационная стоимость связей между вершиной критического пути и всеми его смежными вершинами, не входящими в критический путь.

Переменная p_{im} представляет назначение исполнителя i на работу m, принимая значение 1, если утвержден факт назначения, в противном случае 0.

$p_{imhbz} \geq 0$ для $i, m \in W, E$

Ограничение (1) обозначает, что каждому исполнителю назначена в точности одна задача.

Ограничение (2) обозначает, что для каждой задачи назначен один исполнитель.

Ограничение (3) обозначает, что при составлении перечня задач следует учитывать только сотрудников, чей отпуск и сроки выполнения текущей задачи не пересекаются с датами выполнения назначаемой задачи.

Ограничение (4) обозначает, что при распределении задач требуется учитывать критерии эффективности сотрудников.

Ограничение (5) обозначает, что при распределении поручений требуется учитывать однозначное назначение кураторов каждому исполнителю.

2.6 Описание алгоритма

Алгоритм распределения задач основан на математической модели системы.

Данный алгоритм можно сформулировать в виде следующей последовательности шагов:

1. получить список задач на распределение;
2. получить множество исполнителей-кандидатов;
3. сформировать матрицу назначений;
4. определить оптимальное решение.

Список задач на распределение формируется ответственным пользователем и может корректироваться в режиме реального времени. При этом в обязательном порядке задача создается с указанием приоритетов упорядочивания исполнителем своих задач по приоритету, представленному в таблице 3:

Таблица 3. Порядок приоритета задачи

Порядок приоритета	Описание
Высокий	Задачу необходимо распределить в первую очередь с установлением времени начала её выполнения сразу после окончания распределения.
Средний	Задачу следует выполнить после задач с приоритетом «Высокий», но по возможности исполнителем, не имеющим актуальных задач в момент распределения.
Низкий	Задача распределяется в последнюю очередь, важнейшим фактором является дата внедрения, не выходящая за установленные временные рамки.

Множество исполнителей-кандидатов представляет собой список всех сотрудников компании, независимо от их принадлежности подразделениям, роду решаемых задач и занятости, поскольку распределяются все трудовые ресурсы компании с равномерной нагрузкой в зависимости от рабочих графиков.

При вычислении критериев эффективности сотрудников алгоритм изначально задает три уровня эффективности по всем показателям, представленные в таблице 4:

Таблица 4. Уровни эффективности сотрудника

Уровень эффективности	Описание
База (1)	Исходная точка, от которой отсчитывается результат. Худшее значение.

Уровень эффективности	Описание
Норма (2)	Уровень, который в обязательном порядке должен быть достигнут с учетом всех обстоятельств.
Цель (3)	Уровень, к которому нужно стремиться, своего рода идеальный показатель.

Оцениваются три ключевых критерия: количество, качество, сроки выполнения работы [12]. Для каждого из показателей устанавливается вес в процентном эквиваленте: количество – 35%, качество – 40%, сроки – 25%. Оценка по критериям высчитывается как произведение уровня эффективности на вес показателя, переведенное в проценты [1]. Дополнительными нормализующими значениями являются области специализации сотрудников ($0 < w_1 < 5$) и их сертификаты, подтверждающие квалификацию ($0 < w_2 < 10$).

Для составления матрицы назначений и определения оптимального решения используется алгоритм целочисленного программирования, а именно метод «ветвей и границ» [2].

1. Наименованиями полей назначаются уникальные идентификаторы исполнителей, наименованиями записей – уникальные идентификаторы задач. Матрица инициализируется стоимостными значениями задач.
2. В каждой строке матрицы назначений находится минимальный элемент и вычитается из каждого элемента строки. Тем самым в матрице появится не менее одного элемента, равного нулю.
3. В каждом столбце матрицы назначений находится минимальный элемент и при условии отсутствия в столбце нуля вычитается из элементов столбца.
4. Выбирается пара претендентов (исполнитель - задача) на ветвление, для которых значение ячейки равно нулю.

Рассчитывается коэффициент путем сложения минимального значения элемента строки задачи и минимального значения столбца – исполнителя. Из всех коэффициентов выбирается максимальный, тем самым определено оптимальное решение и задача текущей строки назначается исполнителю текущего поля.

5. Так как каждому исполнителю назначается только одна работа, то удаляем исполнителя и задачу из матрицы.

Описание алгоритма произведено с помощью блок-схем. Использование элементов блок-схем регламентируется ГОСТ 19.701-90 «Единая система программной документации. Схемы алгоритмов, программ, данных и систем.»

Рисунок 7 – Основная блок-схема алгоритма распределения задач

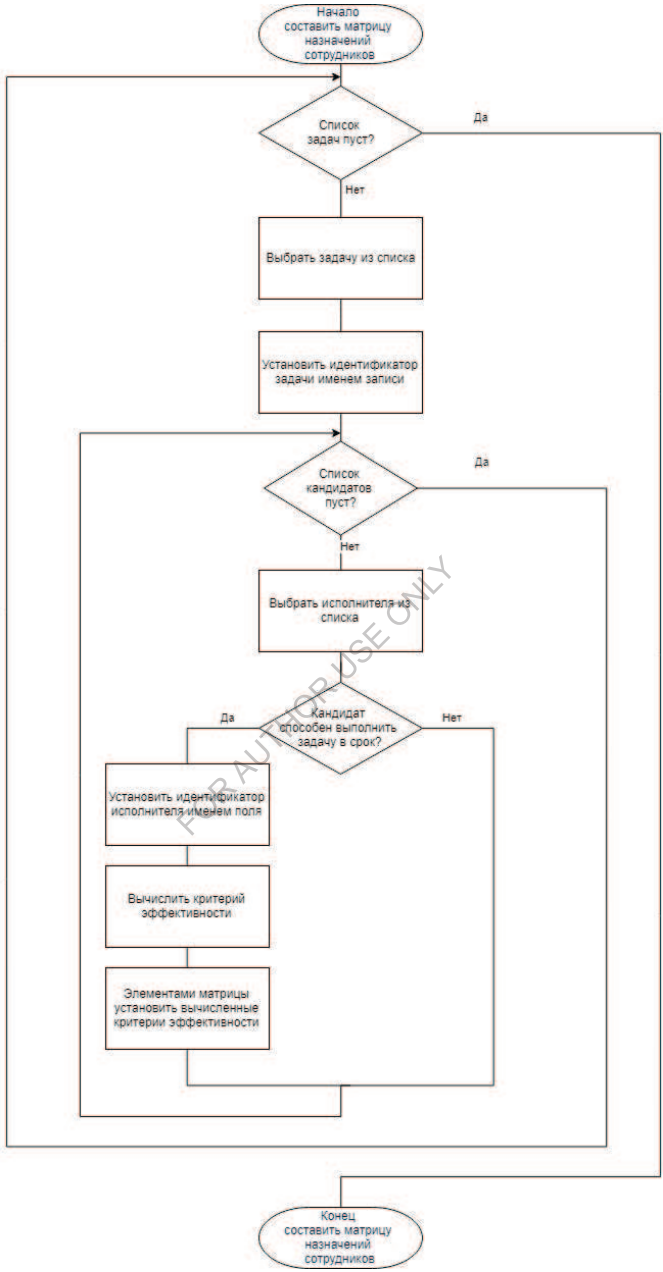


Рисунок 8 – Блок-схема процесса «Составить матрицу назначений сотрудников»

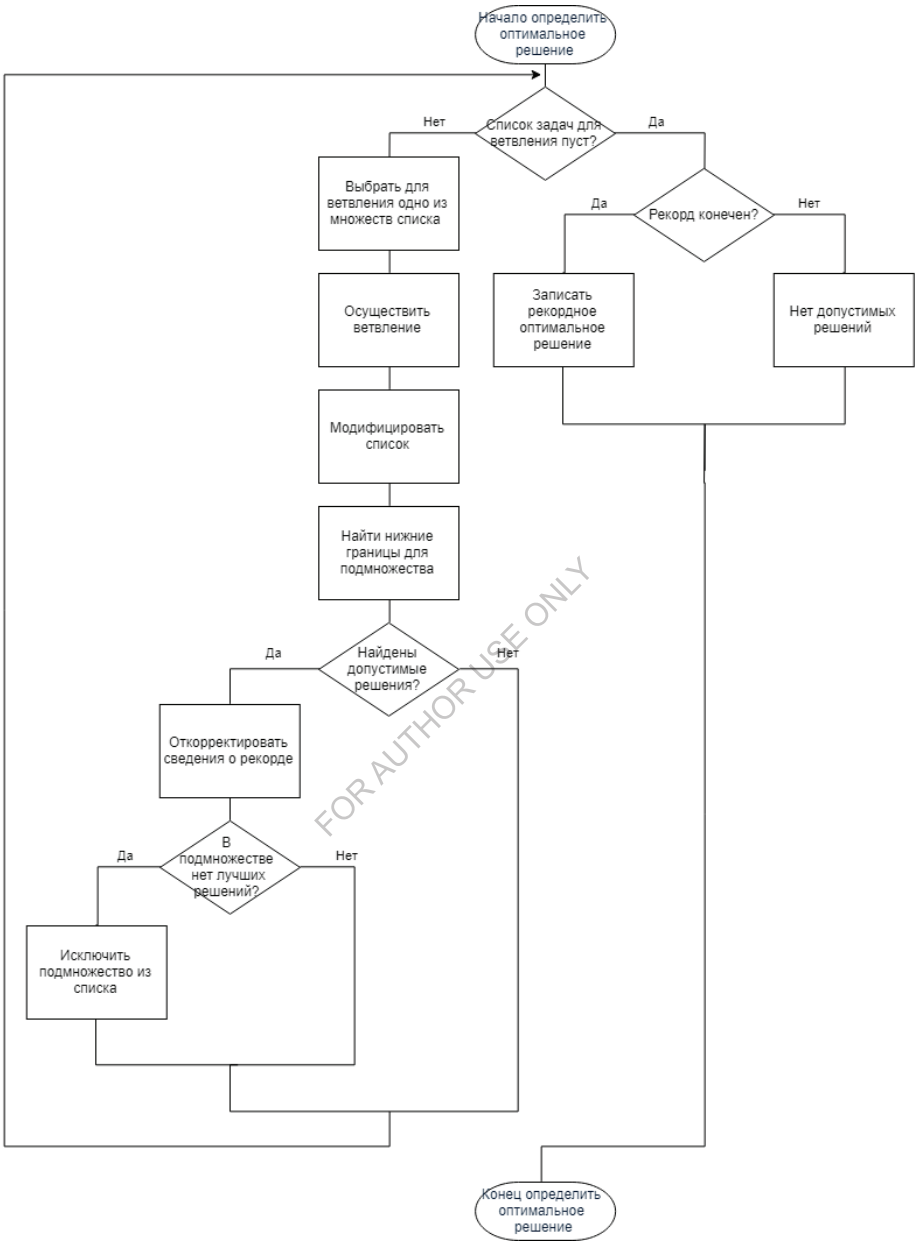


Рисунок 9 – Блок-схема процесса «Определить оптимальное решение»

3 Разработка прототипа программного продукта по автоматизированному распределению задач

3.1 Обоснование среды разработки

На основании составленной математической модели, описанного алгоритма и сформулированного технического задания, приведенных в разделе Разработка математической модели и алгоритма распределения задач, был разработан прототип программного продукта для автоматизации распределения внутренних производственных задач предприятия с учетом специфики предметной области.

Для достижения поставленных целей в рамках исследовательской работы в качестве среды разработки используется платформа «1С:Предприятие». Система программ «1С:Предприятие 8» представляет из себя технологическую платформу и разработанные на ее основе прикладные решения, которые позволяют автоматизировать различные сферы человеческой деятельности [14]. Архитектура «1С:Предприятие» позволяет смешивать в одной системе элементы, работающие под управлением различных операционных систем, что позволяет разрабатывать кроссплатформенные приложения (Рисунок 10) [19][34].

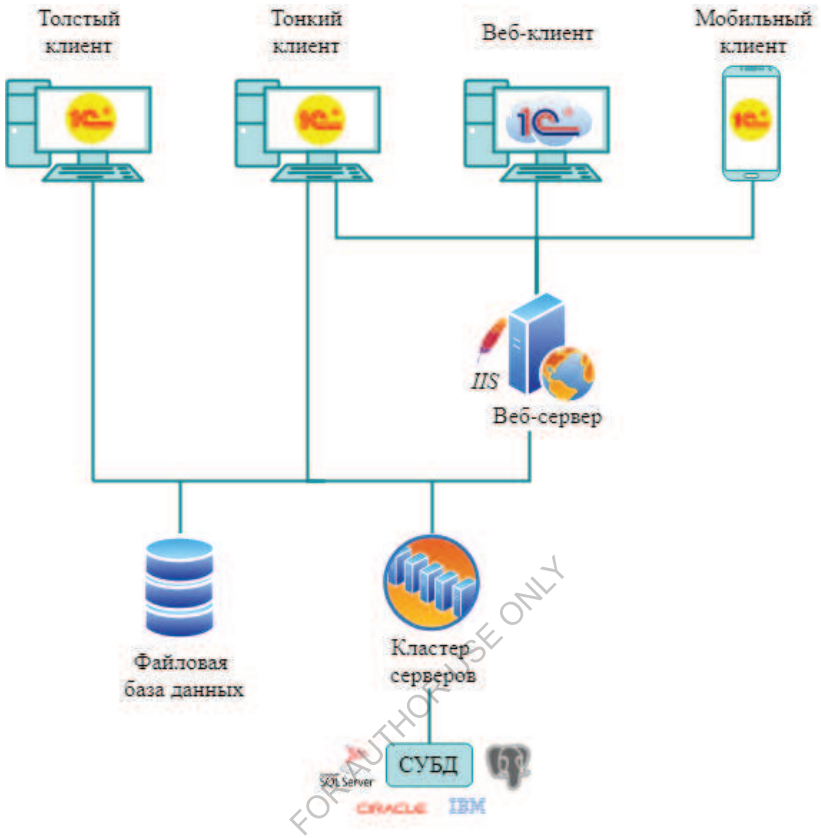


Рисунок 10 – Архитектура платформы «1С:Предприятие»

Классическим вариантом запуска является запуск прикладных решений в файловом или клиент-серверном вариантах работы на персональных компьютерах пользователей под управлением операционной системы Windows, Linux или Mac OS X [15].

Альтернативным способом, набирающим популярность ввиду широкого распространения беспроводной высокоскоростной передачи данных между устройствами в частности мобильной связи, является запуск прикладных решений на портативных устройствах под управлением операционных систем Android и IOS [16].

Создание мобильных приложений возможно за счет наличия мобильной платформы, которая входит в комплект поставки 1С [23]. Разработка осуществляется традиционными средствами разработки 1С на персональном компьютере в режиме конфигуратора, с той лишь разницей, что существуют некоторые ограничения при работе с объектами конфигурации [24]. Разработанное прикладное решение, установленное на устройстве, представляет из себя совокупность мобильной платформы и информационной базы [25].

Веб-клиент - это один из режимов запуска клиентского приложения, не имеющего исполняемого файла [13]. Веб-клиент исполняется в среде интернет-браузера и адаптирован для работы пользователей с прикладным решением через интернет. Для начала работы любому пользователю достаточно запустить браузер и перейти по адресу веб-сервера, на котором опубликована информационная база.

Для работы в режиме веб-клиента требуется веб-сервер Apache или IIS, настроенный на работу с «1С:Предприятие 8». Браузер клиента взаимодействует с веб-сервером по протоколу HTTP или HTTPS. Веб-сервер, в свою очередь, взаимодействует с «1С:Предприятие 8» в файловом или клиент-серверном варианте работы.

3.2 Разработка прикладного решения

3.2.1 Подсистема нормативно-справочной информации

Для хранения нормативно-справочной информации используются такие объекты конфигурации, как справочники. Они позволяют хранить в информационной базе данные, имеющие одинаковую структуру и списочный характер, а также предоставляют другим объектам 1С информацию для обработки.

Справочник «Исполнители»

Данный справочник содержит сведения о всех сотрудниках организации, необходимые для организации автоматизированного распределения задач. В обязательном порядке указываются персональные

данные, такие как фамилия, имя, отчество и область специализации. Сведения о загруженности на текущий момент и значения эффективности работы вычисляются автоматически на основе данных, занесенных в систему.

В табличной части «Отпуск» отражаются данные о согласованных с руководством отпусках.

В табличной части «Уровень квалификации» можно задать список документов, подтверждающих прохождение обучения по подтверждению или повышению квалификации.

Таблица 5. Выдержка из технического проекта. Описание справочника «Исполнители»

№	Реквизит	Тип	Примечание
1	Код	Строка (9)	
2	Наименование	Строка (150)	Ассоциативное наименование
3	Фамилия	Строка (100)	Фамилия исполнителя
4	Имя	Строка (100)	Имя исполнителя
5	Отчество	Строка (100)	Отчество исполнителя (в случае наличия обязательно к заполнению)
6	Область специализации	Перечисление «Типы задач»	Род задач, выполнением которых занимается сотрудник

№	Реквизит	Тип	Примечание
7	Загруженность	Перечисление «Статус готовности сотрудника»	Показатель загруженности сотрудника на текущий момент времени
8	Критерий эффективности	Число (10, 2)	Значение ключевого значения эффективности сотрудника
9	Подразделение	Справочник «Структура организации»	Отдел, за которым документально закреплен сотрудник, напрямую связан с областью специализации
	Табличная часть «Отпуск»		Описывает заявления на отпуск, требующие обязательного согласования с руководством
1	Вид отпуска	Справочник «Виды отпусков»	Значение вида отпуска определяет

№	Реквизит	Тип	Примечание
			назначение и способ оплаты работодателем
2	Дата начала	Дата	Дата начала отпуска
3	Дата окончания	Дата	Дата последнего дня отпуска
	Табличная часть «Уровень квалификации»		Описывает квалификацию каждого сотрудника
1	Имя документа	Строка (150)	Представление документа
2	Номер документа	Строка (20)	Номер документа, подтверждающего квалификацию
3	Дата получения	Дата	Дата получения документа ф формате dd.MM.yyyy
4	Срок действия	Дата	Дата окончания действия документа
5	Бессрочный	Булево	Признак того, что документ действует бессрочно

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 11, при запуске мобильного клиента на рисунке 12.

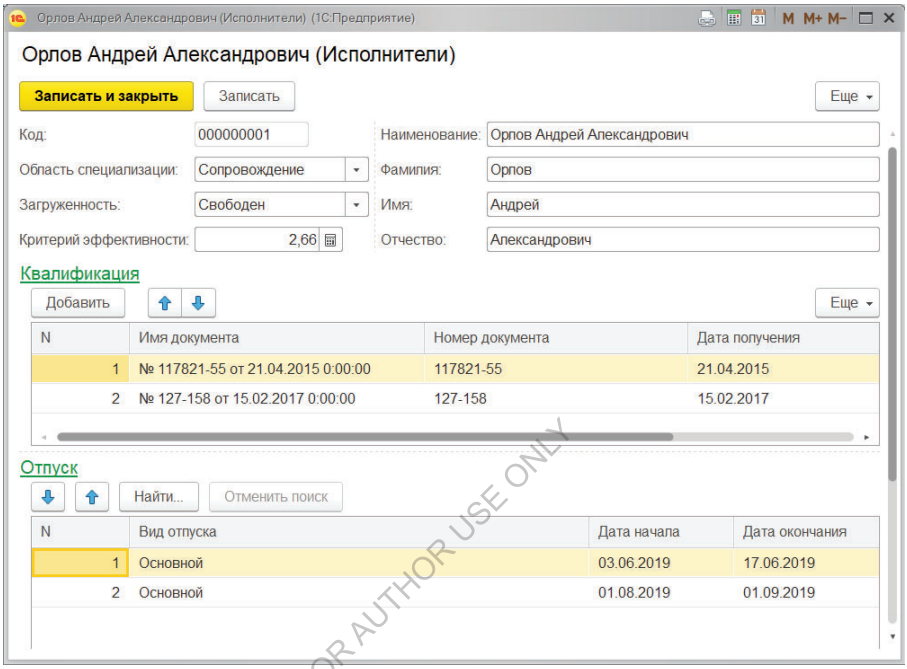


Рисунок 11 - Справочник «Исполнители»

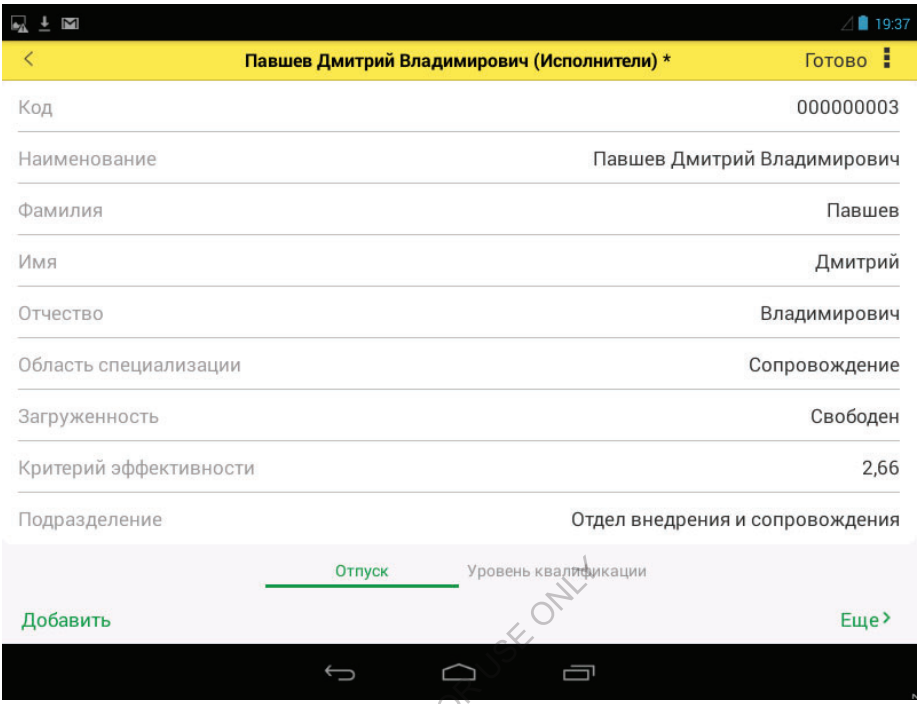


Рисунок 12 – Внешний вид справочника «Исполнители» в мобильном клиенте

Справочник «Задачи»

Данный справочник предназначен для хранения списка задач для делегирования их потенциальным исполнителям. Доступ к справочнику предоставляется только пользователям с ролью «Ответственный».

Таблица 6. Выдержка из технического проекта. Описание справочника «Задачи»

№	Реквизит	Тип	Примечание
1	Код	Строка (9)	
2	Наименование	Строка (150)	Ассоциативное наименование
3	Дата начала	Дата	Дата начала

№	Реквизит	Тип	Примечание
			задачи соответствует дате создания поручения
4	Дата окончания	Дата	Планируемая дата внедрения задачи
5	Тип задачи	Перечисление «Типы задач»	Тип решаемых задач зависит от принадлежности сотрудника подразделению организации
6	Исполнитель задачи	Справочник «Исполнители»	Сотрудник, ответственный за выполнение поручения
7	Уровень сложности	Перечисление «Уровни сложности»	Показатель сложности выполняемой задачи предназначен для назначения только исполнителям с высоким коэффициентом эффективности
8	Куратор	Справочник	Сотрудник,

№	Реквизит	Тип	Примечание
		«Исполнители»	ответственный за выполнение исполнителем поручения
9	Предыдущий исполнитель	Справочник «Исполнители»	Сотрудник, который ранее выполнял задачу
10	Приоритет	Перечисление «Приоритет задачи»	Показатель важности выполнения задачи
11	Наиболее ранняя дата начала	Дата	Наиболее ранняя дата начала выполнения поручения
12	Крайняя дата внедрения	Дата	Наиболее поздняя дата внедрения решения
13	Описание	Строка	Подробная формулировка задания
14	Статус готовности	Перечисление «Статусы готовности задачи»	Показатель уровня готовности назначенного поручения: от уровня «Не готова» до

№	Реквизит	Тип	Примечание
			«Сдана»
15	Дата сдачи	Дата	Реальная дата сдачи выполненной работы. Предназначена для определения отклонения от поставленных сроков

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 13, при запуске мобильного клиента на рисунке 14.

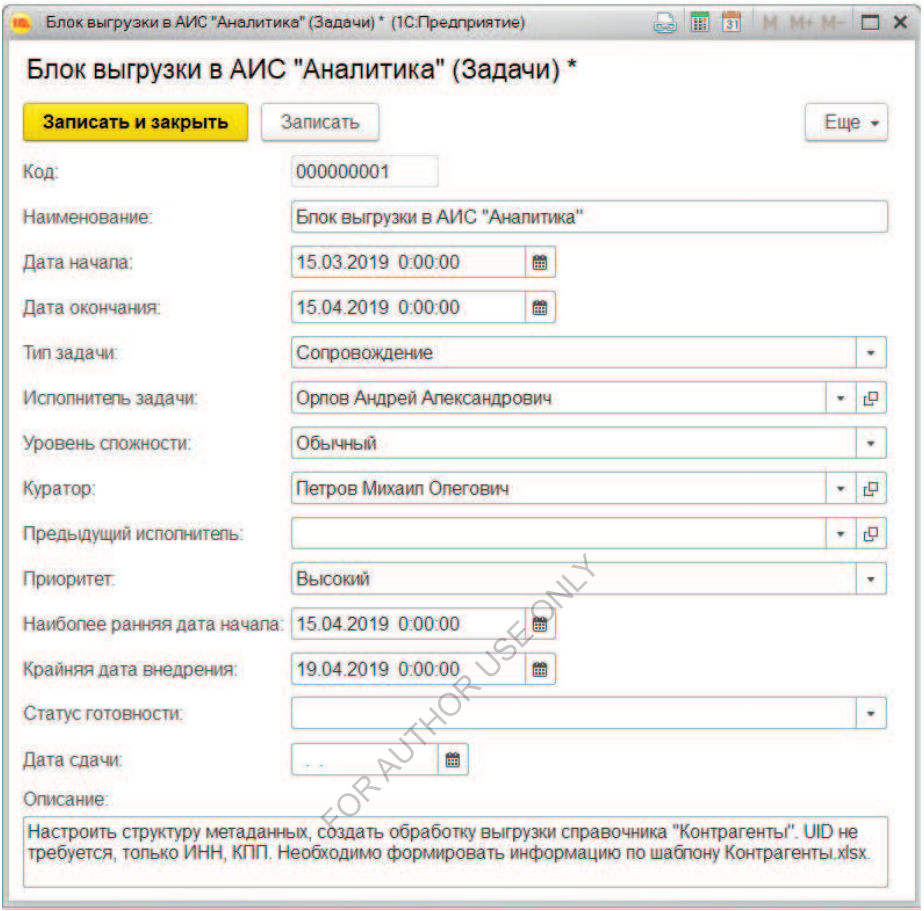


Рисунок 13 - Справочник «Задачи»

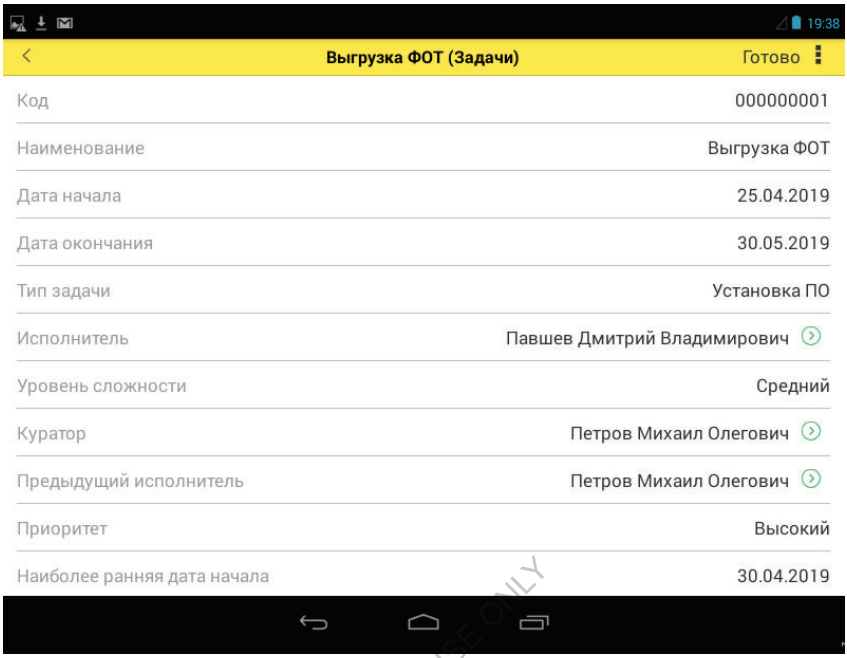


Рисунок 14 – Внешний вид справочника «Задачи» в мобильном клиенте

Справочник «Пользователи»

Данный справочник дублирует список пользователей, доступный из режима конфигуратора (Рисунок 15). Создание нового пользователя возможно нажатием на кнопку «Записать» или «Записать и закрыть», при этом программно создается новый пользователь системы с назначенной ролью. В случае попытки записать уже существующего пользователя перезаписывается пароль существующего элемента. Следует отметить, что работа с этим справочником доступна только пользователям с правами администратора.

Таблица 7. Выдержка из технического проекта. Описание справочника «Пользователи»

№	Реквизит	Тип	Примечание
1	Код	Строка (9)	

№	Реквизит	Тип	Примечание
2	Наименование	Строка (50)	Имя и полное имя пользователя
3	Пароль	Дата	Комбинация символов, содержащая строчные и заглавные символы латинского алфавита, цифры и специальные знаки
4	Роль	Справочник «Роли»	Роль пользователя, указание которой обязательно при создании нового пользователя для определения прав на работу с объектами

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 16.

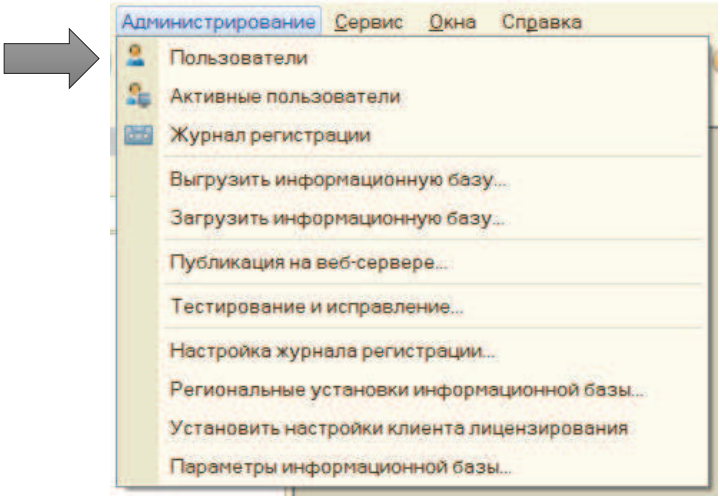


Рисунок 15 – Иерархическое меню для создания пользователя

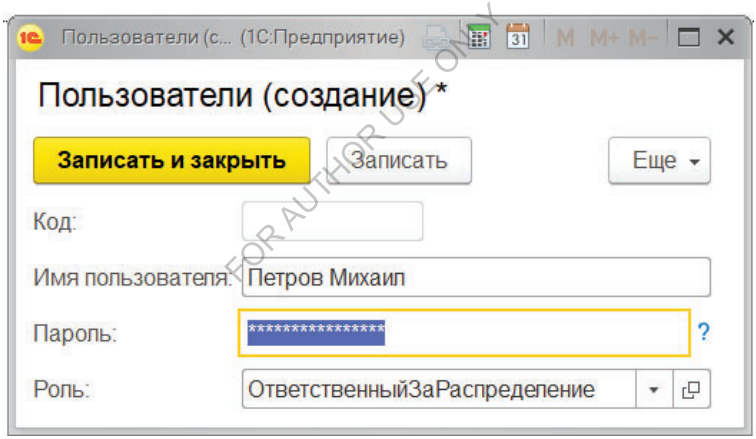


Рисунок 16 - Справочник «Пользователи»

Справочник «Виды отпусков»

Данный справочник содержит перечень видов отпусков для использования в зависимости от назначения. Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 17.

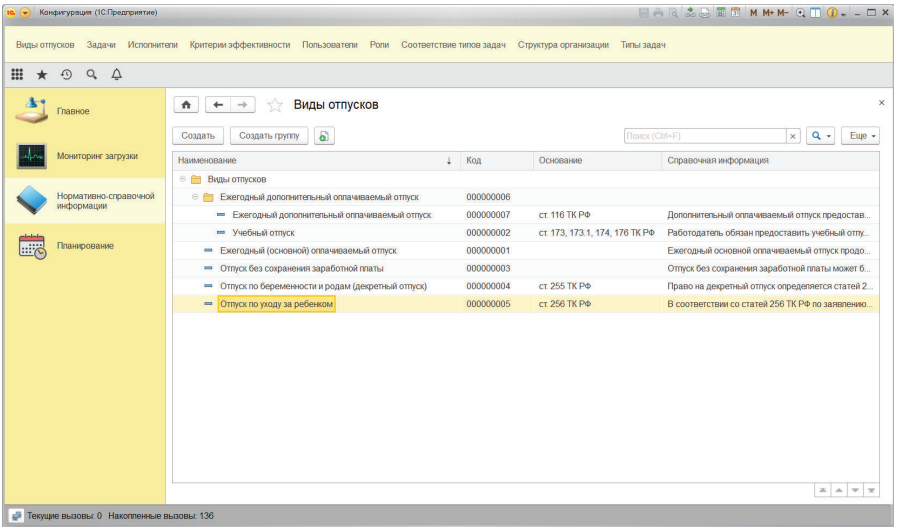


Рисунок 17 - Справочник «Виды отпусков»

Справочник «Графики работы сотрудников»

Данный справочник содержит список и настройку графиков работы персонала с учетом выходных и праздничных дней.

Таблица 8. Выдержка из технического проекта. Описание справочника «Графики работы сотрудников»

№	Реквизит	Тип	Примечание
1	Код	Строка (9)	
2	Наименование	Строка (25)	Ассоциативное наименование
3	Год отображаемого графика	Число (4, 0)	Год для формирования рабочих графиков
4	Среднемесячное число часов	Число (10, 5)	Вычисляемое число рабочих часов в месяц

№	Реквизит	Тип	Примечание
5	Среднемесячное число дней	Число (10, 5)	Вычисляемое число рабочих дней в месяц
6	Неполный рабочий день	Булево	Показатель полного или неполного рабочего дня
7	Неполная рабочая неделя	Булево	Показатель неполной рабочей недели

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 18.

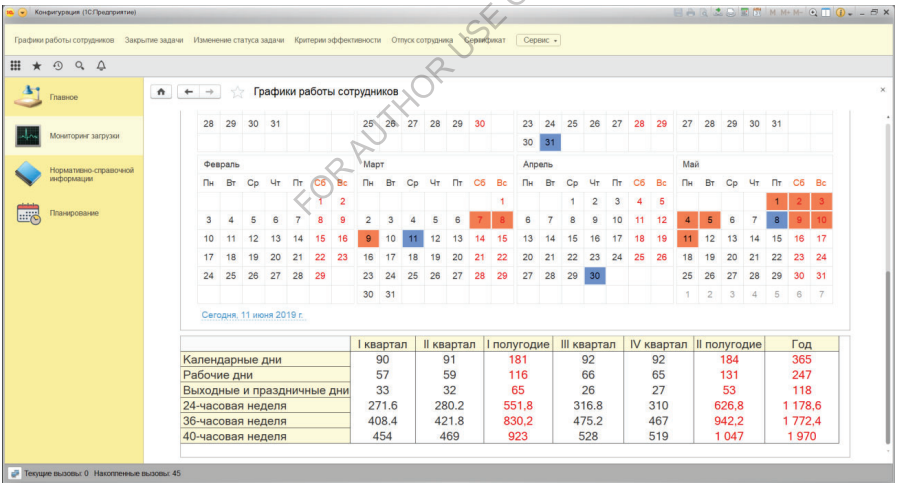


Рисунок 18 - Справочник «Графики работы сотрудников»

Справочник «Критерии эффективности»

Данный справочник содержит основные критерии эффективности выполнения задач сотрудников, список несет в себе лишь справочную

информацию, не содержит никакой смысловой нагрузки и влияния на результат оценивания персонала. Внешний вид справочника в разработанном прикладном решении представлен на рисунке 19.

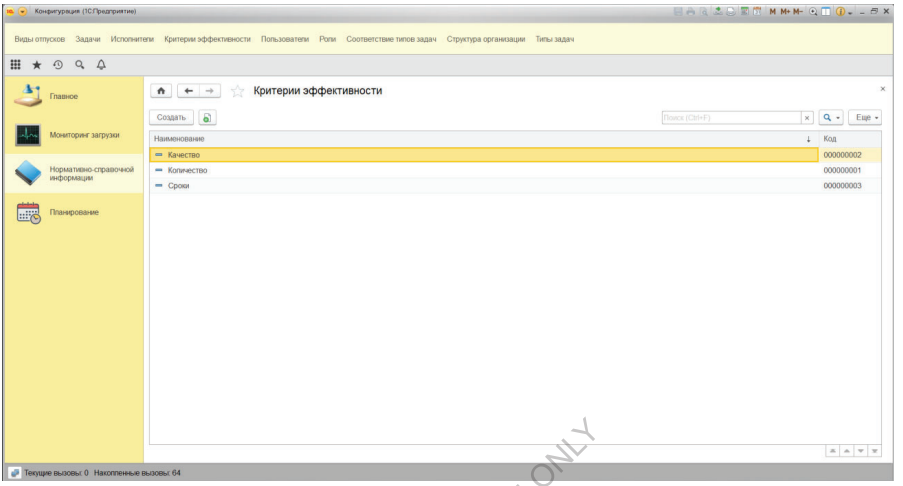


Рисунок 19 - Справочник «Критерии эффективности»

Документ «Отпуск сотрудников»

Данный документ отражает заявку на отпуск, которая попадает в перечень документов на согласование с руководством.

Таблица 9. Выдержка из технического проекта. Описание документа «Отпуск сотрудников»

№	Реквизит	Тип	Примечание
1	Номер	Строка (9)	Номер документа, присваиваемый по умолчанию
2	Дата	Дата	Дата согласования документа
3	Руководитель	Справочник	Лицо,

№	Реквизит	Тип	Примечание
		«Сотрудники»	выполняющее функции руководителя организации
4	Сотрудник	Справочник «Сотрудники»	Лицо, подающее заявление на отпуск
5	Дата начала	Дата	Дата начала отпуска
6	Дата окончания	Дата	Дата окончания отпуска
7	Вид отпуска	Справочник «Виды отпусков»	Значение вида отпуска определяет назначение и способ оплаты работодателем

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 20.

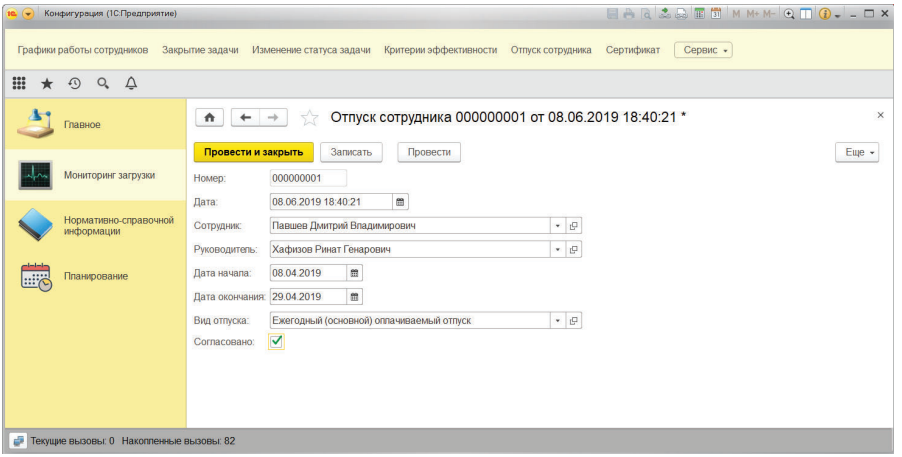


Рисунок 20 - Документ «Отпуск сотрудников»

Документ «Сертификаты»

Данный документ содержит информацию об имеющихся сертификатах, дипломах и удостоверениях по профилю работы в отделе.

Таблица 10. Выдержка из технического проекта. Описание документа «Сертификаты»

№	Реквизит	Тип	Примечание
1	Номер	Строка (9)	Уникальный номер документа
2	Дата	Дата	Дата начала действия документа
3	Сотрудник	Справочник «Сотрудники»	Имя владельца документа
4	Имя документа	Строка (100)	Имя документа, подтверждающего квалификации, включающее в

№	Реквизит	Тип	Примечание
			себя номер и дату выдачи
5	Срок действия	Дата	Дата окончания действия документа
6	Бессрочный	Булево	Показатель неограниченного срока действия документа

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 21.

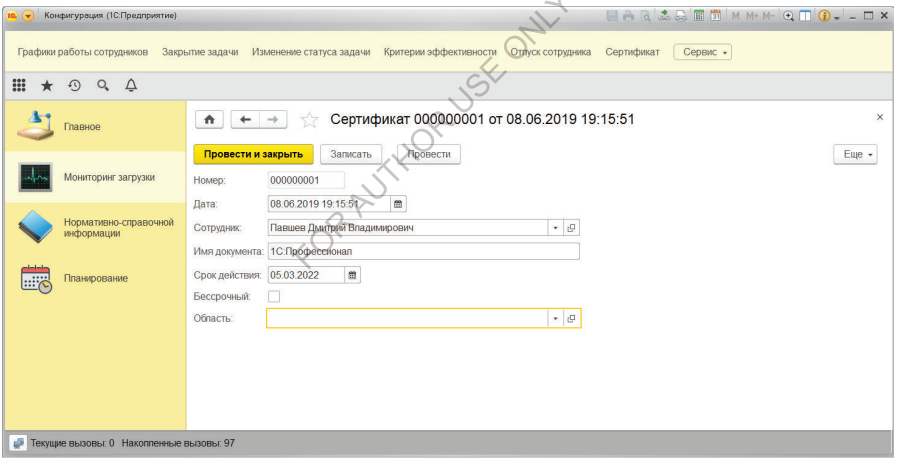


Рисунок 21 - Документ «Сертификаты»

Документ «Закрытие задачи»

Данный документ предназначен для закрытия работ вне зависимости от исполнителя, при этом возможность закрытия имеет только куратор задачи.

Таблица 11. Выдержка из технического проекта. Описание документа «Закрытие задачи»

№	Реквизит	Тип	Примечание
1	Номер	Строка (9)	Номер документа, присваиваемый по умолчанию
2	Дата	Дата	Дата проведения документа
3	Задача	Справочник «Задачи»	Представление задачи из общего списка
4	Дата сдачи	Дата	Дата закрытия задачи

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 22.

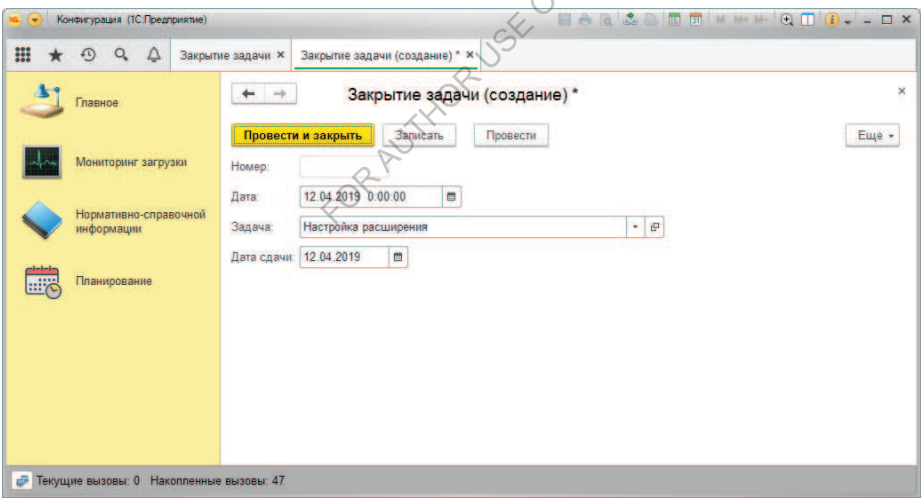


Рисунок 22 - Документ «Закрытие задачи»

Документ «Изменение статуса задачи»

Данный документ предназначен для изменения статуса задачи, при этом возможность работы с данным свойством задачи имеют только

сотрудники, имеющие непосредственное отношение к работе, то есть исполнитель и куратор задачи.

Таблица 12. Выдержка из технического проекта. Описание документа «Изменение статуса задачи»

№	Реквизит	Тип	Примечание
1	Номер	Строка (9)	Номер документа, присваиваемый по умолчанию
2	Дата	Дата	Дата проведения документа
3	Задача	Справочник «Задачи»	Представление задачи из общего списка
4	Статус	Перечисление «Статусы готовности задачи»	Текущий статус выполнения задачи

Внешний вид справочника в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 23.

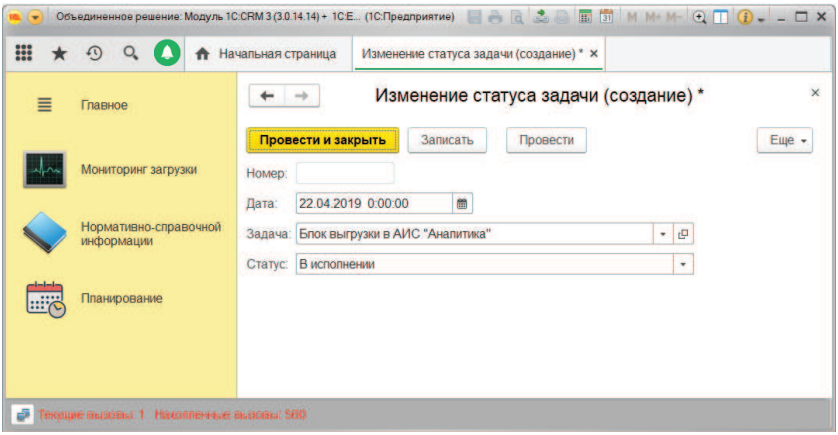


Рисунок 23 - Документ «Сертификаты»

3.2.2 Подсистема мониторинга загрузки

Обработка «Отчет о загруженности сотрудников»

Содержит информацию о загрузке по процессам с отображением динамики исполнения заданий [6]. Поскольку возможности мобильной платформы по работе с отчетами ограничены, создана универсальная обработка, которую можно использовать в любом из режимов запуска прикладного решения [20][21]. Внешний вид отчета в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 24, в мобильном решении на рисунке 25.

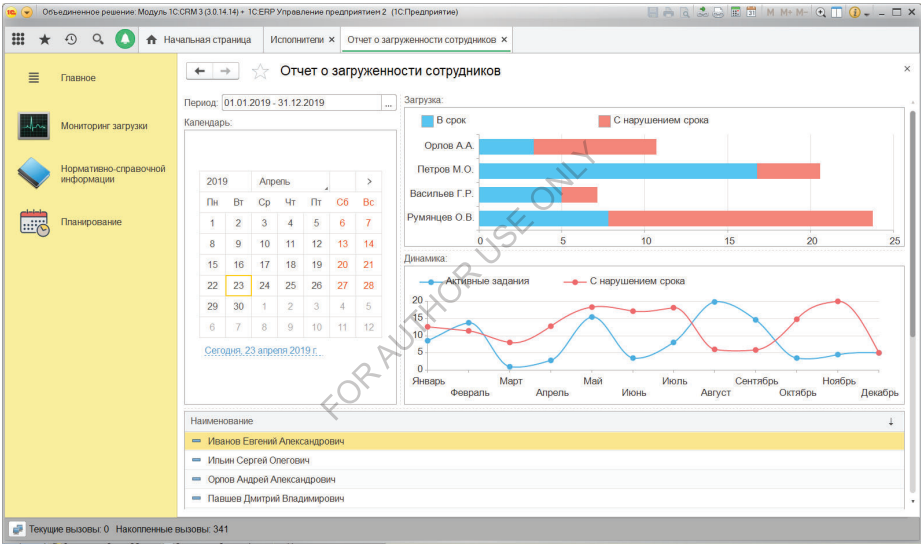


Рисунок 24 - Обработка «Отчет о загруженности сотрудников»

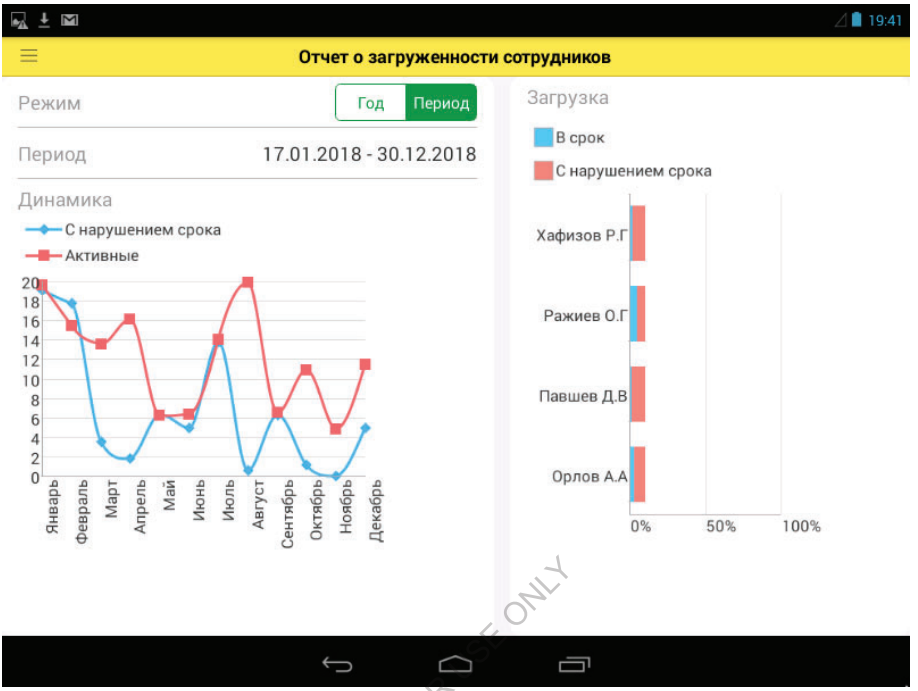


Рисунок 25 - Отчет о загруженности сотрудников в мобильном клиенте

Обработка «Отчет об эффективности сотрудников»

Данная обработка содержит информацию о динамике исполнительской дисциплины сотрудников организации за годовой период, крайним месяцем является текущий [7]. Внешний вид отчета в разработанном прикладном решении для запуска на персональном компьютере представлен на рисунке 26, в мобильном клиенте на рисунке 27.

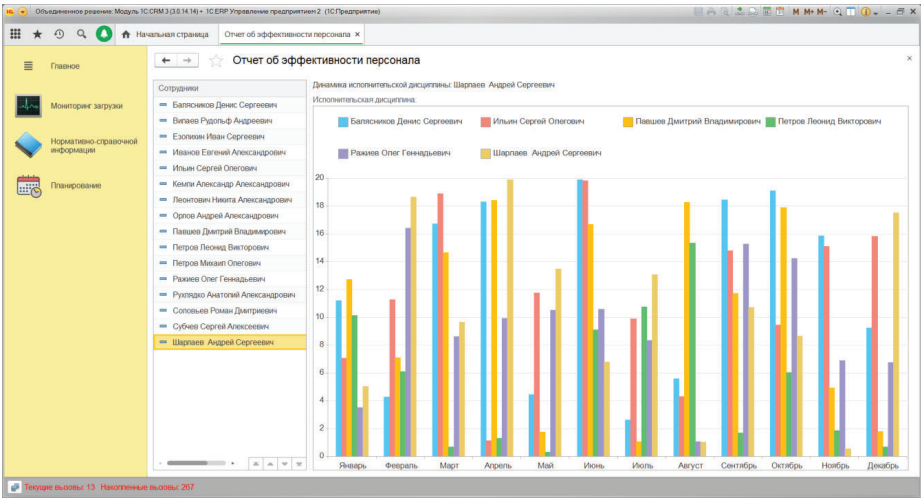


Рисунок 26 - Обработка «Отчет об эффективности сотрудников»

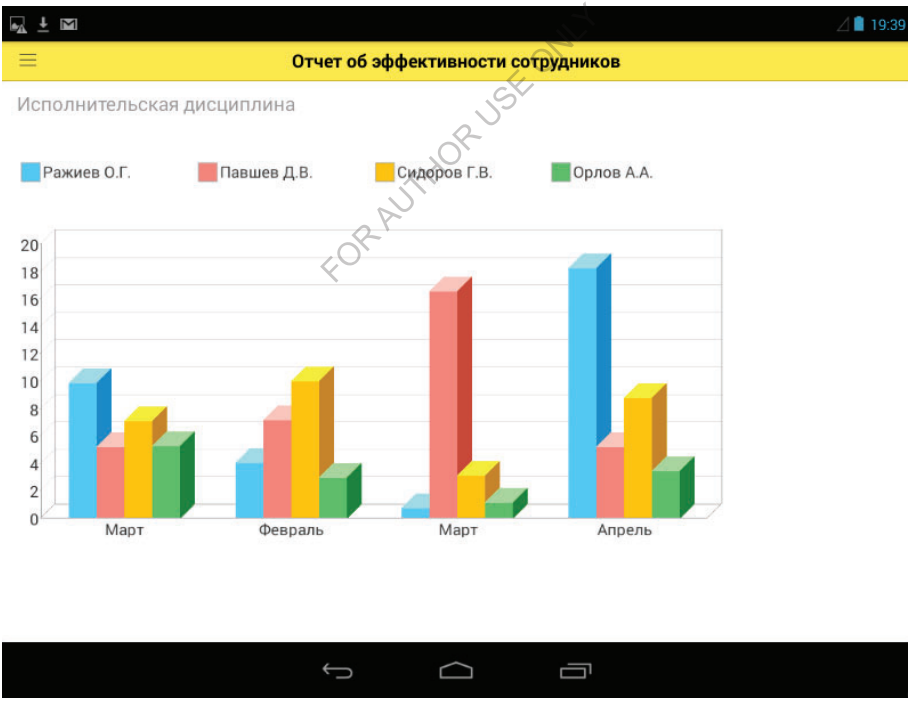


Рисунок 27 - Отчет об эффективности сотрудников в мобильном клиенте

Обработка «Мои задачи»

Данная обработка предоставляет пользователю возможность просматривать и отслеживать актуальные и завершенные задачи, а также процесс их выполнения.

Таблица 13. Выдержка из технического проекта. Описание обработки «Мои задачи»

№	Реквизит	Тип	Примечание
1	Код	Строка (9)	
2	Сотрудник	Справочники «Сотрудники»	Имя исполнителя задач. Устанавливается автоматически в
			зависимости от активного пользователя системы
	Табличная часть «Активные задачи»		
1	Имя	Строка (150)	Наименование актуальной задачи
2	Дата внедрения	Дата	Крайняя дата внедрения, срок более которой будет свидетельствовать о нарушении сроки и назначению штрафов

№	Реквизит	Тип	Примечание
3	Куратор	Строка (150)	Ответственный за выполнения задачи
	Табличная часть «Завершенные задачи»		Отображает список завершенных задач для конкретного пользователя системы
1	Имя	Строка (150)	Наименование завершенной задачи
2	Дата сдачи	Дата	Дата проведения документа о закрытии задачи

Внешний вид обработки для формирования перечня завершенных и актуальных задач представлен на рисунке 28.

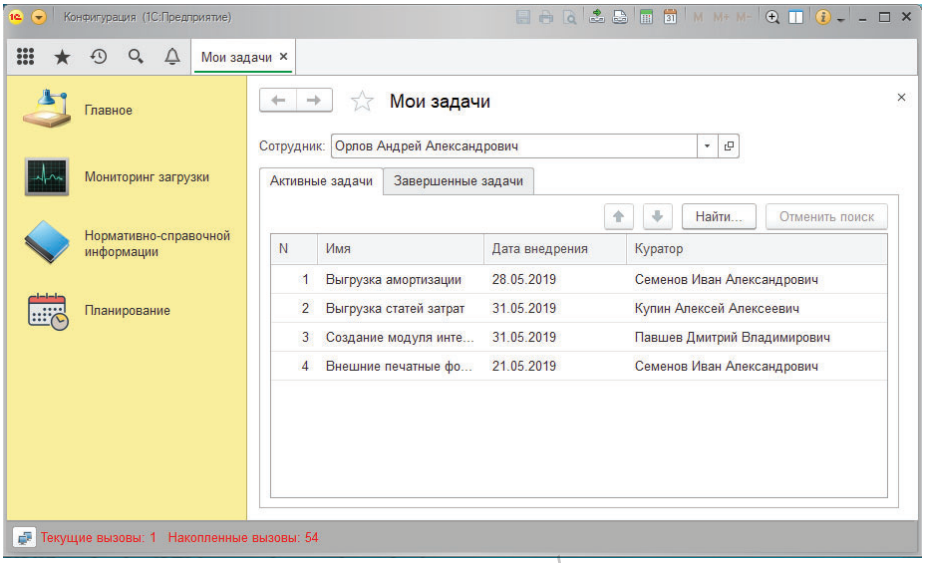


Рисунок 28 – Обработка «Мои задачи»

3.2.3 Подсистема планирования

Обработка «Обмен с мобильным приложением»

Поскольку мобильное приложение (мобильный узел) и приложение для запуска с персонального компьютера (центральный узел) неразрывно связаны друг с другом и должны регулярно обмениваться информацией создана обработка, запуск которой осуществляется на мобильном устройстве под управлением операционной системы Android.

Обмен данными между мобильным и центральным узлом осуществляется через веб-сервисы: мобильный клиент вызывает веб-сервисы, развернутые на стороне центрального узла. Механизм обмена представлен в нотации UML, унифицированного языка моделирования, на рисунке 29.

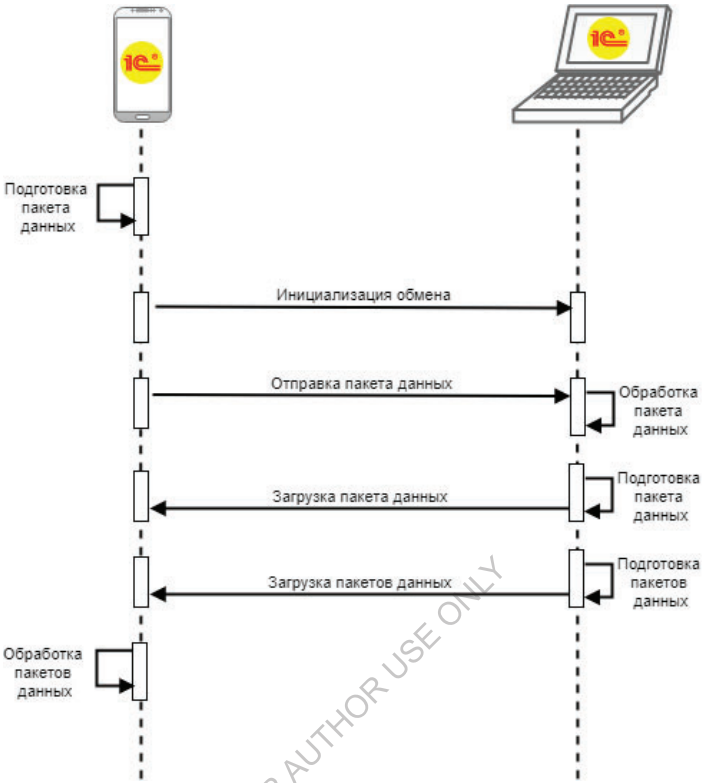


Рисунок 29 – Диаграмма последовательности механизма обмена данными

Мобильное приложение инициирует обмен данными, первоначально формируя пакет обмена, содержащий идентификатор мобильного приложения и данные, обновленные на мобильном устройстве со момента последней синхронизации, и пересылает его в центральный узел [32]. Получив информацию из стартового пакета, центральный узел формирует и упаковывает данные в пакеты формата XDTO, измененные в центральном узле с момента последней синхронизации. Пакеты XDTO позволяют описать структуру передаваемых данных для преобразования в XML и из XML. Однако размер каждого пакета ограничен содержание не более 500 объектов

[33]. Регистрация изменений одного объекта представлена в нотации UML на рисунке 30.

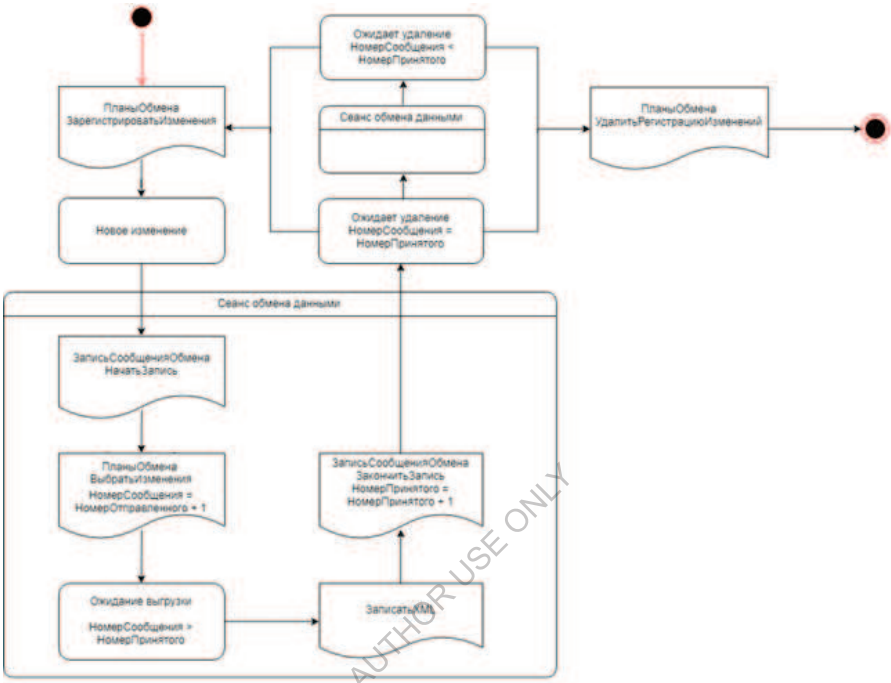


Рисунок 30 – Диаграмма состояний регистрации изменения одного объекта данных

Мобильный узел последовательно получает пакеты от центрального узла. После загрузки последнего пакета мобильный клиент начинает обрабатывать полученные данные, то есть совершать запись и проведение в зависимости от объектов обмена.

Интерфейс мобильного приложения на странице обмена данными между основным и мобильным клиентом представлен на рисунке 31.

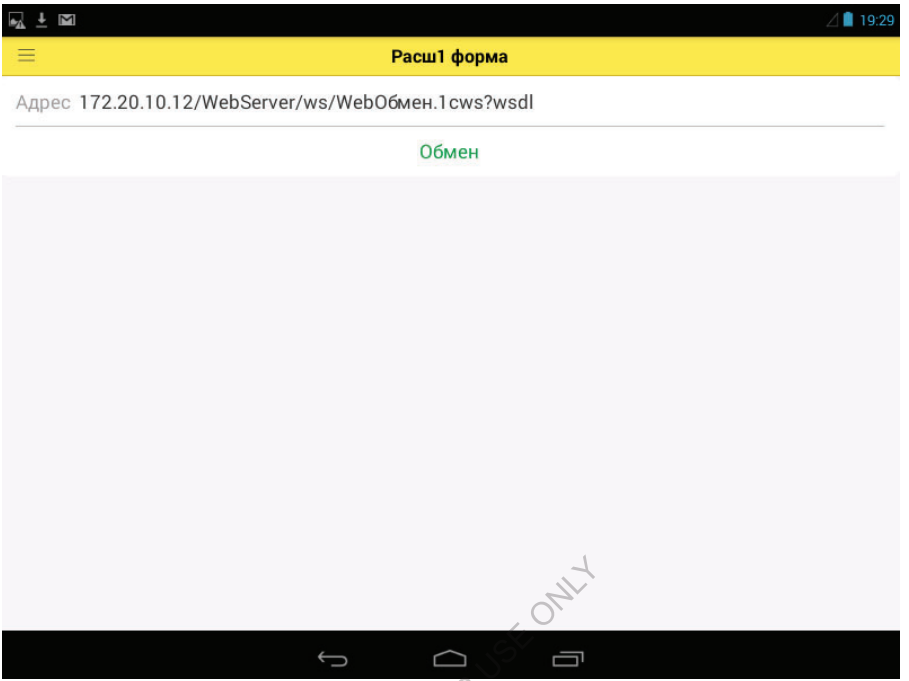


Рисунок 31 – Интерфейс обработки для обмена данными на мобильном устройстве

Для осуществления запуска мобильного клиента необходимо осуществить публикацию на веб-сервере. Данная встроенная возможность платформы используется в двух случаях: запуск решения в режиме веб-клиента, запуск мобильного решения. Опубликовать разработанное решение можно путем открытия диалогового окна (Администрирование – Публикация на веб-сервере) [31]. Диалоговое окно публикации на веб-сервере представлено на рисунке 32.

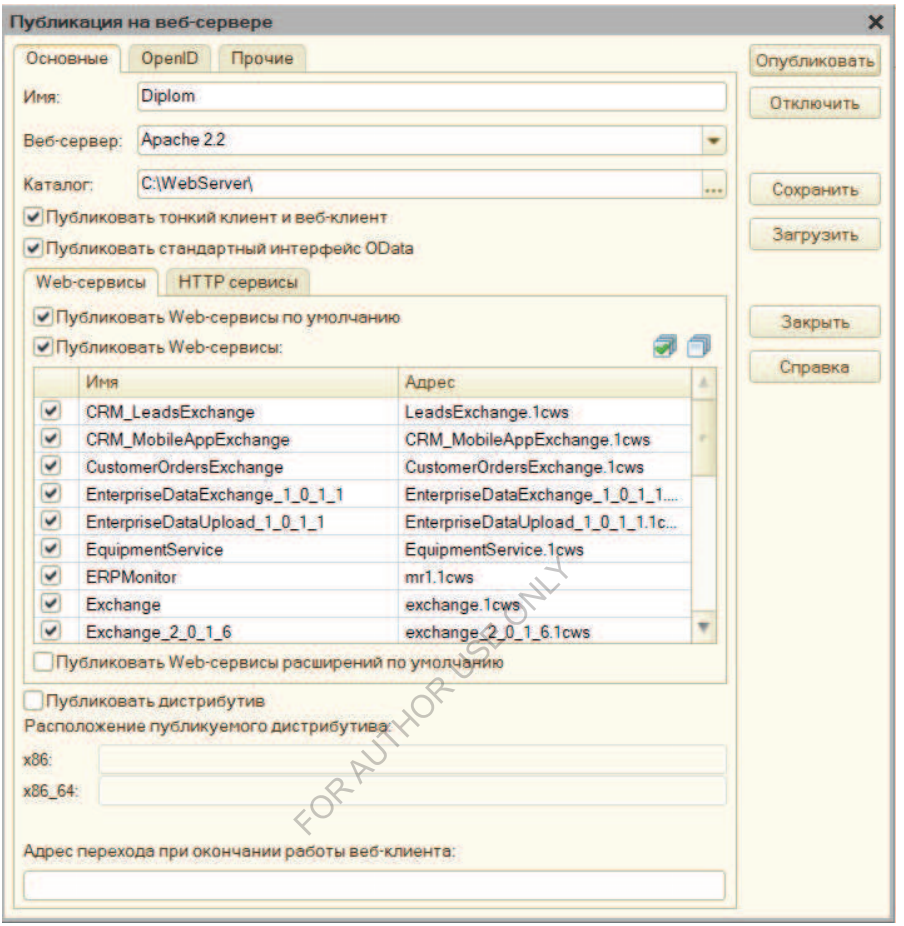


Рисунок 32 – Публикация на веб-сервере

Проверку опубликованного приложения можно осуществить путем перехода в браузере по пути «localhost/Diplom», указав имя публикации.

Публикация мобильного приложения используется для добавления новой конфигурации на мобильном устройстве и незначительно отличается от публикации обычного приложения.

В первую очередь следует открыть диалоговое окно публикации (Конфигурация – Мобильное приложение – Публиковать или Конфигурация

– Мобильное приложение – Обновить публикуемое приложение, если публикация выполнялась ранее). Диалоговое окно публикации мобильного приложения представлено на рисунке 33.

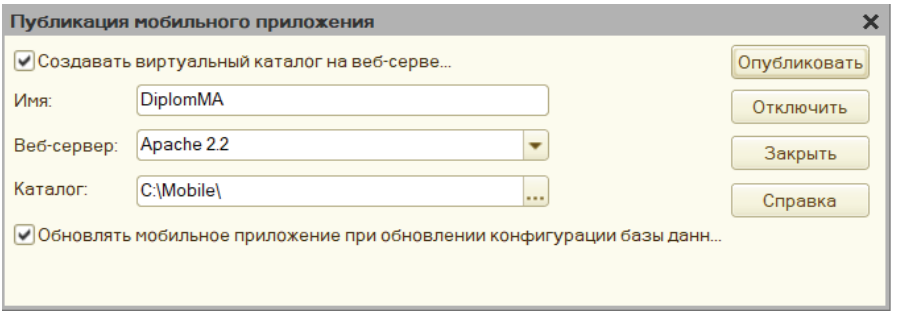


Рисунок 33 – Диалоговое окно публикации мобильного приложения

Для создания виртуального каталога на веб-сервере требуется указать имя и выбрать веб-сервер. Для веб-сервера IIS можно указать использование NTLM аутентификации.

В поле «Каталог» ввести путь к каталогу, в котором располагаются файлы описания виртуального каталога. В случае использования веб-сервера Apache имя каталога должно содержать только символы латинского алфавита.

По кнопке «Опубликовать» происходит:

- создание каталога на веб-сервере;
- создание каталога на диске;
- предлагается размещение в нем мобильного приложения;
- происходит проверка актуальности информационной базы;
- в случае неактуальности базы происходит обновление информационной базы;
- проверка информационной базы;

- если ошибок нет, выполняется выгрузка информационной базы в мобильное приложение, в противном случае выгрузка не производится.

Кнопка «Отключить» выполняет отмену публикации и удаляет каталог веб-сервера.

Опубликованное мобильное приложение представляет собой XML-файл, который можно получить, перейдя по ссылке «localhost/DiplomMA» в браузере или открыв файл в каталоге публикации на диске.

Обработка «Производственный календарь»

Объект предназначен для отображения списка поручений на месяц или на конкретную дату для выбранного пользователя.

Таблица 14. Выдержка из технического проекта. Описание обработки «Производственный календарь»

№	Реквизит	Тип	Примечание
1	Сотрудник	Справочник «Исполнители»	Реквизит для получения списка задач конкретного пользователя
2	Календарь	Дата	Стандартный период для формирования списка задач
3	Описание задачи	Строка	Подробное описание выбранной задачи из таблицы значений

№	Реквизит	Тип	Примечание
	Таблица значений «Задачи на месяц»		Объект содержит перечень задач на текущий месяц задачи
1	Наименование	Строка (150)	Наименование
2	Дата окончания	Дата	Срок исполнения задачи

Интерфейс формы производственного календаря представлен на рисунке 34.

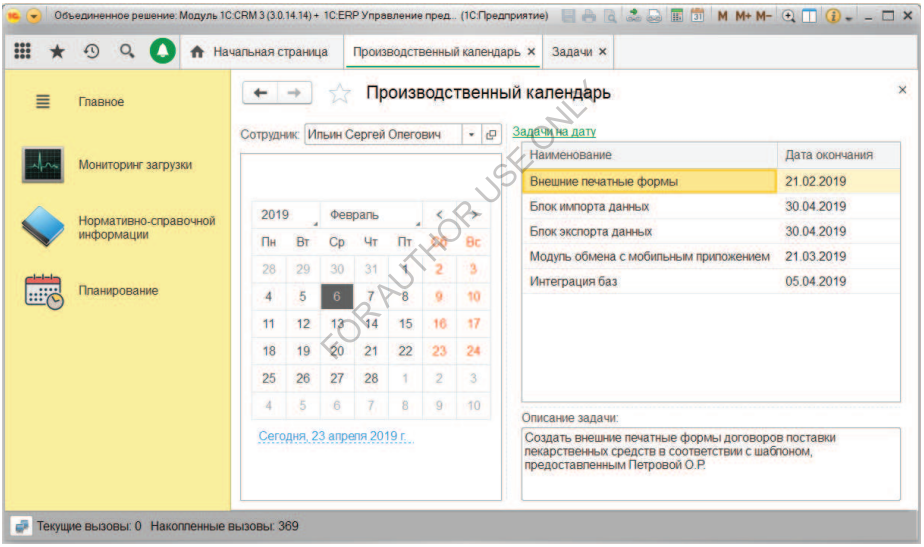


Рисунок 34 - Обработка «Производственный календарь»

Регистр сведений «Критерии эффективности»

Данные регистр содержит информацию для расчета индивидуальных критериев эффективности в соответствии с составленной математической моделью. Внешний вид объекта представлен на рисунке 35.

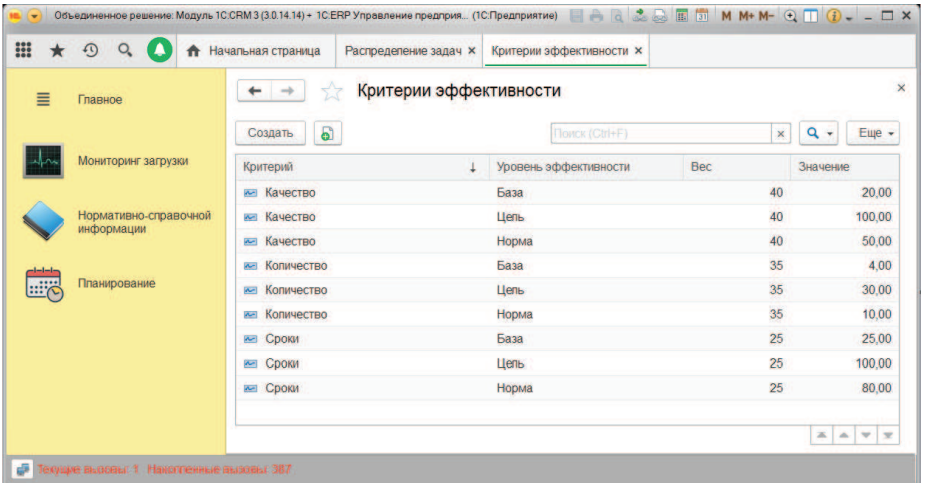
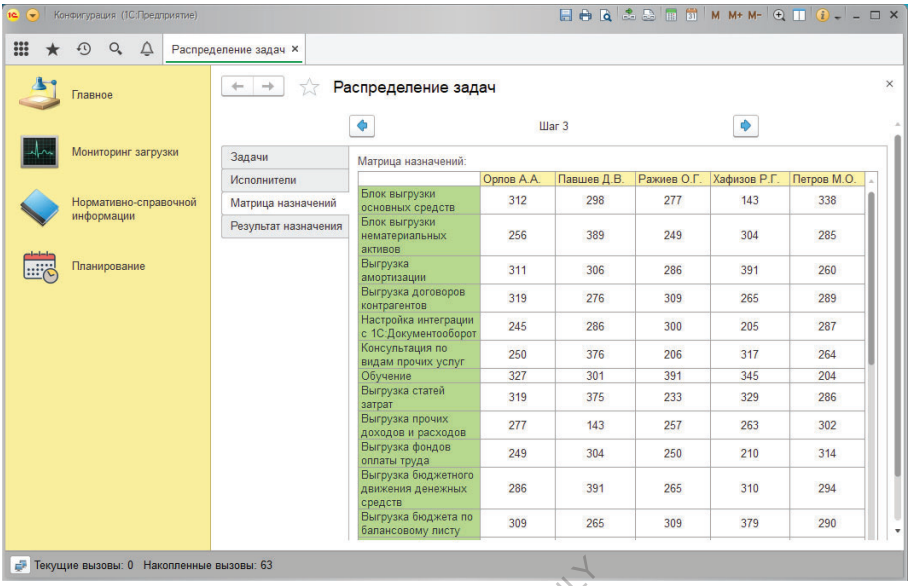


Рисунок 35 - Регистр сведений «Критерии эффективности»

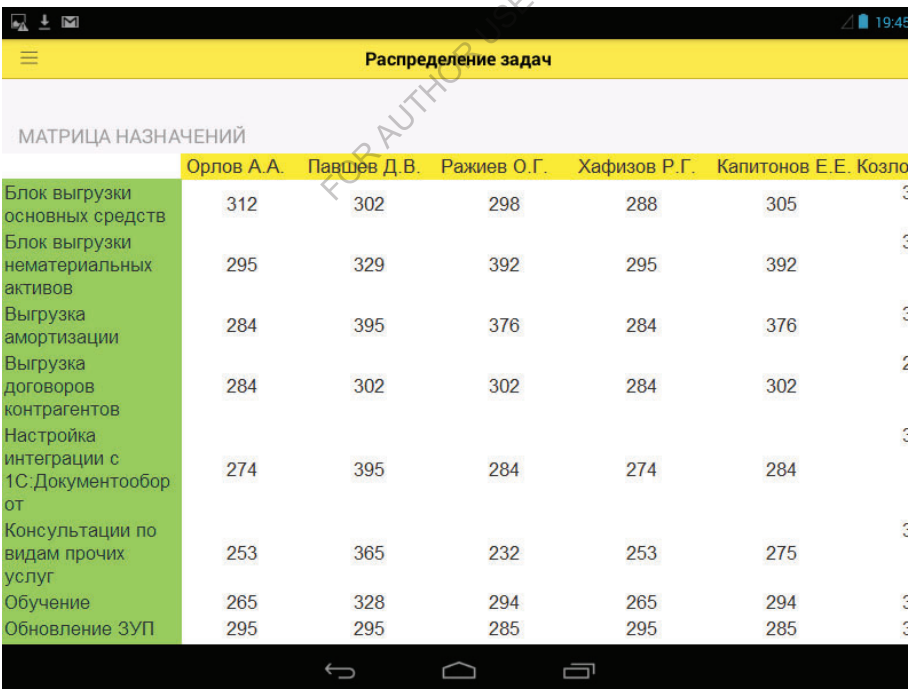
Обработка «Распределение задач»

В данной обработке реализован основной механизм автоматизированного распределения задач. На рисунке 36 представлена сформированная матрица назначений, на основе которой происходит расчет в решении на персональном компьютере, рисунок 37 дублирует форму в мобильном клиенте. Итоговый результат распределения представлен на рисунке 38.



	Орлов А.А.	Павшев Д.В.	Ражиев О.Г.	Хафизов Р.Г.	Петров М.О.
Блок выгрузки основных средств	312	298	277	143	338
Блок выгрузки нематериальных активов	256	389	249	304	285
Выгрузка амортизации	311	306	286	391	260
Выгрузка договоров контрагентов	319	276	309	265	289
Настройка интеграции с 1С:Документооборот	245	286	300	205	287
Консультация по видам прочих услуг	250	376	206	317	264
Обучение	327	301	391	345	204
Выгрузка статей затрат	319	375	233	329	286
Выгрузка прочих доходов и расходов	277	143	257	263	302
Выгрузка фондов оплаты труда	249	304	250	210	314
Выгрузка бюджетного движения денежных средств	286	391	265	310	294
Выгрузка бюджета по балансовому листу	309	265	309	379	290

Рисунок 36 – Матрица назначений



	Орлов А.А.	Павшев Д.В.	Ражиев О.Г.	Хафизов Р.Г.	Капитонов Е.Е.	Козло
Блок выгрузки основных средств	312	302	298	288	305	
Блок выгрузки нематериальных активов	295	329	392	295	392	
Выгрузка амортизации	284	395	376	284	376	
Выгрузка договоров контрагентов	284	302	302	284	302	
Настройка интеграции с 1С:Документооборот	274	395	284	274	284	
Консультации по видам прочих услуг	253	365	232	253	275	
Обучение	265	328	294	265	294	
Обновление ЗУП	295	295	285	295	285	

Рисунок 37 – Матрица назначений в мобильном клиенте

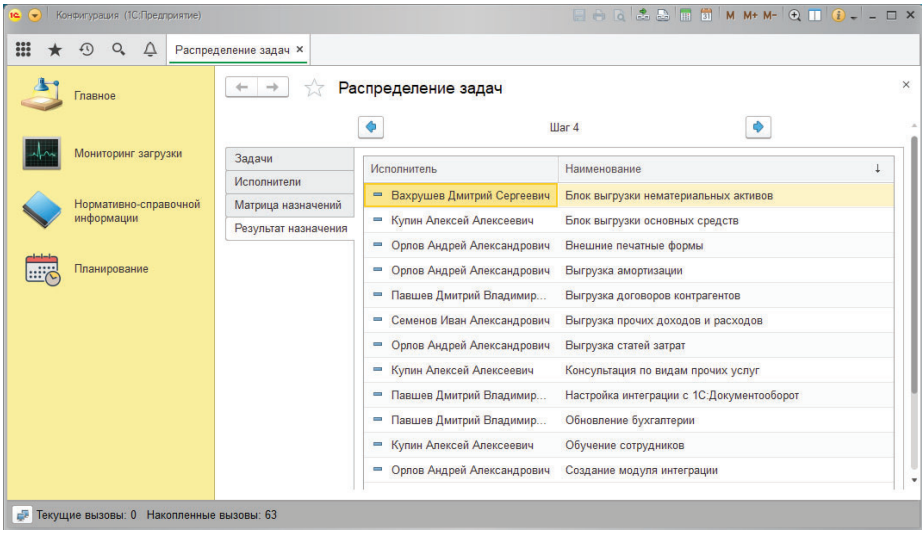


Рисунок 38 – Результат распределения задач

4 Тестирование системы

Для проведения сценарного тестирования было разработано прикладное решение, использующее встроенные механизмы платформы. Для написания тестовых сценариев проверки версии программного продукта для запуска на персональном компьютере использовались механизмы автоматизированного тестирования, которые недоступны в мобильном клиенте [17].

Автоматизированное тестирование представляет из себя взаимодействие двух информационных баз: клиент тестирования и менеджер тестирования [18]. Менеджер тестирования осуществляет подключение к клиенту и проводит интерактивное воспроизведение реальных действий пользователя в соответствии с прописанными программно [26].

Для тестирования мобильного приложения использовалась обработка, встроенная в прикладное решение, которая показывала результаты, соизмеримые с результатами автоматизированного тестирования на ПК [22].

Определение режима запуска осуществляется при создании и редактировании информационной базы (Рисунок 39) или в режиме конфигуратора в параметрах запуска «1С:Предприятие» (Рисунок 40). Данное дополнительное условие является обязательным для осуществления сценарного тестирования.

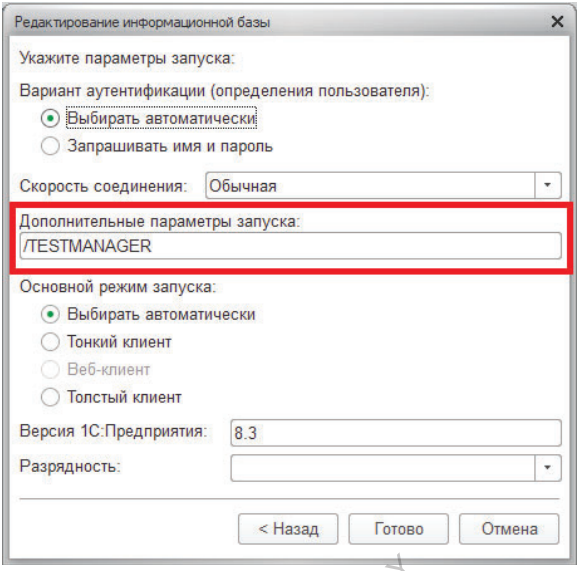


Рисунок 39 – Изменение параметров запуска «1С:Предприятие»

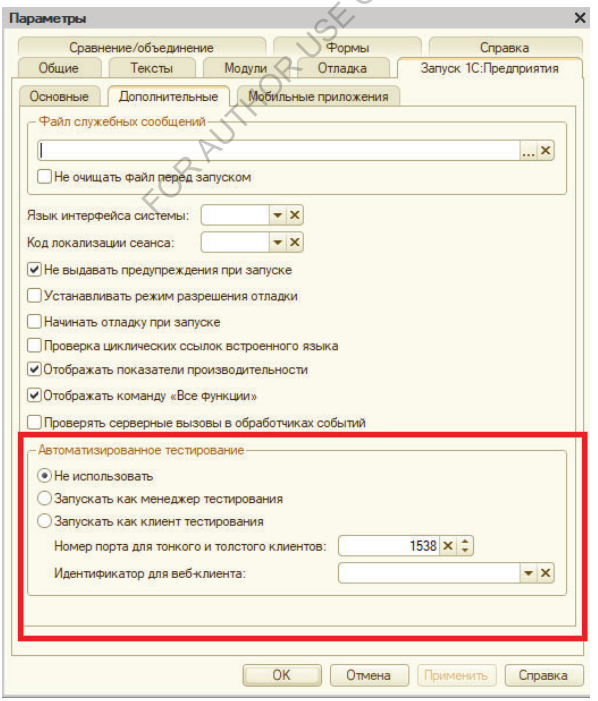


Рисунок 40 – Изменение свойства запуска «1С:Предприятие»

В рамках исследовательской работы клиентом тестирования является разработанное программное обеспечения для автоматизированного распределения задач.

Менеджер тестирования предоставляет возможность запуска сценариев, представленных в таблице 15. Сценарии являются идентичными в отношении работы с объектами конфигурации, разница заключается в количестве создаваемых элементов справочников и проведении документов, что позволяет оценить работу прикладного решения в условиях высокой нагрузки со стороны пользователей.

Таблица 15. Тестовые сценарии

Наименование	Количество исполнителей	Количество документов «Сертификаты» у исполнителя	Количество задач	Количество документов «Изменение статуса задачи»
TC_01	100	5	100	100
TC_02	500	10	1000	100
TC_03	1000	15	10000	200

Сценарное тестирование позволяет выявить ошибки на этапе работы пользователя с данными. В результате проведения сценарного тестирования были выявлены следующие ошибки:

- Длина вводимых поля «Наименование» в некоторых случаях превышает допустимые значения. Данная возможность обусловлена максимальной длиной стандартного реквизита, ограниченной 150 символами, на уровне платформы. Методом решения является введение дополнительного поля «Полное наименование», а в некоторых случаях конкретная формулировка названия объекта с вынесением дополнительной информации в соответствующие поля, такие как «Описание».

- После проведения документа о закрытии задачи нет возможности отмены совершенного действия и возврата в один из статусов задачи, свидетельствующем о незавершенности. Для устранения этой проблемы решено ввести дополнительный механизм подтверждения закрытия работ пользователям с ролью «Ответственный».

С целью комплексной оценки производительности платформы «1С:Предприятие 8» и определения поведения системы в реальных условиях при различных по уровню и по продолжительности нагрузках проведено нагрузочное тестирование (тестирование производительности). В данном случае сценарное тестирование является вспомогательной частью нагрузочного [27].

Для тестирования на 64-разрядные ОС со всеми последними обновлениями была установлена платформа 8.3.13.1809, платформа устанавливалась с разрядностью x64 на Linux и x32 + x64 на Windows, но ввиду одинаковых результатов с x32 приведено среднее значение всех запусков [28][29]. В качестве конечных результатов приведено среднее значение от трех запусков тестов в таблице 16 и на рисунке 41.

Таблица 16. Результаты тестов

	Windows 7	Windows 10	Ubuntu 18.04	Debian 9.7	Android 6.1	Android 8.1
TC_01	00:07	00:04	00:02	00:03	00:29	00:19
TC_02	00:21	0:15	00:11	0:12	1:47	00:44
TC_03	00:42	00:35	00:27	00:29	2:07	01:22

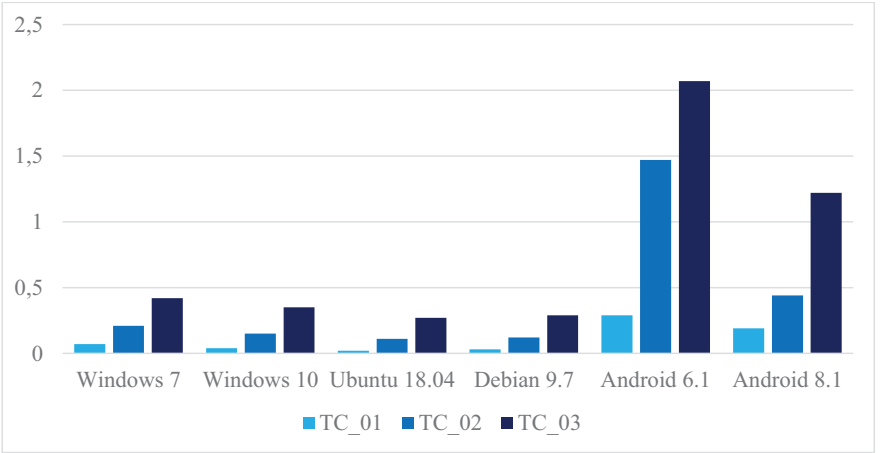


Рисунок 41 – Производительность операционных систем

Среди операционных систем персональных компьютеров ОС Windows 7 показала самые низкий результат. Windows 10 оказалась более производительной, по сравнению с Windows 7 конечный результат оценки превосходит его на 20%, что является значительно высоким показателем [30].

Ведущие современные системы из семейства ОС Linux с мощными графическими оболочками, Ubuntu 18.04 и Debian 9.7, показали результат, превышающий значение Windows 10 на 25%. Это всё говорит о том, что ОС на базе ядра Linux способны работать с системой 1С ничуть не хуже ОС семейства Windows.

Если оценивать производительность мобильной платформы, то операции на мобильном устройстве занимают значительно больше времени, чем на персональном компьютере. Показатели последней версии Android 8.1 показали результаты в среднем в 4 раза ниже ОС Windows 7, поэтому использовать мобильное решение рекомендуется использовать на портативных устройствах с последней версией операционной системы и только при отсутствии доступа к персональному компьютеру, а также сотрудникам, которым периодически требуется выезд к клиентам, исходя из специфики работы.

Заключение

Целью исследовательской работы являлась разработка системы управления с автоматизированным процессом распределения задач путем эффективного использования трудовых ресурсов.

Для достижения поставленной цели были изучены подходы к управлению организациями, разновидности структур управления и типы производственных структур предприятия, что позволило сформировать четкое техническое задание для разработки автоматизированной системы.

Определены и рассчитаны универсальные критерии оценки эффективности деятельности сотрудников любого предприятия вне зависимости от внутренней организации и специфики предметной области.

Изучены существующие алгоритмы решения задач комбинаторной оптимизации в области математической оптимизации или исследовании операций. Составлен алгоритм автоматизированного распределения поручений на примере тестовых данных. По результатам тестирования удалось определить рекомендуемую аппаратную и программную часть средств запуска используемой платформы, выполнены корректировки и доработки прикладного решения, касающие логики программы.

Разработана структура прикладного решения на базе платформы «1С:Предприятие 8». Разработано, протестировано и передано на внедрение кроссплатформенное приложение, автоматизирующее процесс эффективного распределения трудовых ресурсов предприятия.

Список использованных источников

1. **Новиков Ф.А.** Дискретная математика для программистов: учебное пособие для вузов. – СПб.: Питер, 2003.
2. **Соловьев В.И.** Методы оптимальных решений: учебное пособие. – М. Финансовый университет, 2012.
3. **Клюев И.П., Приходько В.А., Трофименко И.В.** Распределение задач между сотрудниками, ориентированными на процедуры или новые возможности, как инструмент развития организации // Технико-экономический вестник «Русского Алюминия». 2012. №22.
4. **Метрики эффективности распределения человеческих ресурсов** – [Электронный источник]. URL: http://miit.ru/contentМетрики%20эффективности%20распределения%20человеческих%20ресурсов.pdf?id_wm=764755
5. **Тютхина А.А.** Методы дискретной оптимизации: Часть 1: Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2014
6. **Постановка заданий сотрудникам разного уровня** – [Электронный источник]. URL: https://marketing.wikireading.ru/39830
7. **Исполнительская дисциплина как ключевой показатель эффективности** – [Электронный источник]. URL: https://www.sekretariat.ru/article/211183-ispolnitelskaya-distsiplina-pokazatel-effektivnosti-17-m12
8. **Перечень требований трудового законодательства** – [Электронный источник]. URL: https://онлайнинспекция.рф/requirements
9. **Конституция Российской Федерации от 12.12.1993) (с учетом поправок, внесенных Законами РФ о поправках к Конституции РФ от 30.12.2008 N 6-ФКЗ, от 30.12.2008 N 7-ФКЗ, от 05.02.2014 N 2-ФКЗ, от 21.07.2014 N 11-ФКЗ)** – [Электронный источник]. URL: http://www.consultant.ru/document/cons_doc_LAW_28399/

10. **Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 27.12.2018)** – [Электронный источник]. URL: http://www.consultant.ru/document/cons_doc_LAW_34683/
11. **Приказ Минздравсоцразвития РФ от 13.08.2009 N 588н "Об утверждении Порядка исчисления нормы рабочего времени на определенные календарные периоды времени (месяц, квартал, год) в зависимости от установленной продолжительности рабочего времени в неделю" (Зарегистрировано в Минюсте РФ 28.09.2009 N 14900)** – [Электронный источник]. URL: http://www.consultant.ru/document/cons_doc_LAW_92167/a47abe7162ac502a7d7e253908900038d89071f6/
12. **Система KPI (Key Performance Indicator): разработка и применение показателей бизнес-процесса. Показатели эффективности** – [Электронный источник]. URL: https://www.businessstudio.ru/articles/article/sistema_kpi_key_performance_indicator_razrabotka_i/
13. **Веб-клиент** – [Электронный источник]. URL: http://v8.1c.ru/overview/Term_000000125.htm
14. **Клиентское приложение** – [Электронный источник]. URL: http://v8.1c.ru/overview/Term_000000121.htm
15. **Многоплатформенность** – [Электронный источник]. URL: http://v8.1c.ru/overview/Term_000000666.htm
16. **Варианты работы системы** – [Электронный источник]. URL: http://v8.1c.ru/overview/Term_000000035.htm
17. **Корпоративный инструментальный пакет** – [Электронный источник]. URL: https://its.1c.ru/db/kip
18. **Сценарное тестирование в помощь программисту 1С** – [Электронный источник]. URL: https://habr.com/ru/post/307808/

19. **Радченко М.Г., Хрусталева Е.Ю.** 1С:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы. - Москва «1С-Паблишинг» 2013
20. **Хрусталева Е.Ю.** Разработка сложных отчетов в «1С:Предприятии 8».Система компоновки данных - Москва «1С-Паблишинг» 2012
21. **Хрусталева Е.Ю.** Язык запросов «1С:Предприятия 8» - Москва «1С-Паблишинг» 2013
22. **Тест-центр. Обзор основных возможностей и принципов работы** – [Электронный источник]. URL: http://v8.1c.ru/expert/tc/tc_overview.htm
23. **Новая версия 8.3 платформы «1С:Предприятие»** – [Электронный источник]. URL: https://docplayer.ru/66129184-Novaya-versiya-8-3-platformy-1s-predpriyatie-peredovaya-oblachnaya-korporativnaya-mobilnaya-krossplatformennaya-i-ne-tolko.html
24. **Хрусталева Е.Ю.** Знакомство с разработкой мобильных приложений на платформе «1С:Предприятие 8» - Москва «1С-Паблишинг» 2015
25. **В. В. Рыбалка.** «Mobile 1С. Пример быстрой разработки мобильного приложения на платформе 1С:Предприятие 8.3. Мастер-класс (+epub)» - Москва «1С-Паблишинг» 2014
26. **Автоматизированное тестирование в «1С:Предприятие 8.3»** – [Электронный источник]. URL: https://курсы-по-1с.рф/news/автоматизированное-тестирование-в-1с-8-3/
27. **Производительность мобильной платформы** – [Электронный источник]. URL: https://курсы-по-1с.рф/articles/скорость-мобильной-платформы/
28. **Нагрузочное тестирование** – [Электронный источник]. URL: http://www.artwell.ru/services/nagruzochnoe-testirovanie/
29. **Тестируем производительность файлового режима 1С:Предприятие в Windows и Linux** – [Электронный источник].

URL: https://interface31.ru/tech_it/2019/01/testiruem-proizvoditel-nost-faylovogo-rezhima-1spredpriyatie-v-windows-i-linux.html

30. **Скорость работы 1С: Предприятие с разными** 340**: MS SQL и с PostgreSQL** – [Электронный источник]. URL: https://infostart.ru/public/995903/
31. **Системные требования 1С:Предприятия 8** – [Электронный источник]. URL: http://v8.1c.ru/requirements/
32. **История одного приложения: мобильное «1С: Управление нашей фирмой»** – [Электронный источник]. URL: https://habr.com/company/1c/blog/331644/
33. **Планы обмена. Квитировать или гарантировать?** – [Электронный источник]. URL: https://infostart.ru/public/567052/
34. **Система управления базами данных** – [Электронный источник]. URL: http://v8.1c.ru/overview/Term_000000662.htm

Приложение 1. Текст программы

```
// Алгоритм распределения задач
&НаКлиенте
Процедура СтраницыПриСменеСтраницы(Элемент, ТекущаяСтраница)
	Если ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница1
		Тогда Шаг = "Шаг 1";
		// Получение списка актуальных задач
		ПолучитьСписокЗадач();

	ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница2
		Тогда Шаг = "Шаг 2";
		// Получение списка исполнителей
		ПолучитьСписокИсполнителей();

	ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница3
		Тогда Шаг = "Шаг 3";

		// Расчет критериев назначения
		Столб = 2;
		Стрк = 2;
		СчетчикИсполнителей = 1;

		Для Каждого Задача Из ЗадачиНаРаспределение Цикл
			Для Каждого Исполнитель Из Исполнители Цикл

				ЗначениеЯчейкиДляРасчета = 0;
				НормализующееЗначение1 = 0;
				НормализующееЗначение2 = 0;
				РасчетноеКачество = 0;
				РасчетныеСроки = 0;
				РасчетноеКачество = 0;

				// Расчет первого нормализующего значения
				НормализующееЗначение1 =
РасчетПервогоПараметра(Исполнитель.ОбластьСпециализации, Задача.ТипЗадачи);

				// Расчет второго нормализующего значения
				НормализующееЗначение2 =
РасчетВторогоПараметра(Исполнитель.Наименование, Исполнитель.ОбластьСпециализации,
Задача.ТипЗадачи);
				Область = МатрицаНазначений.Область(1, Столб, 1, Столб);
				Область.Текст = Исполнитель.Наименование;
				Область.ШиринаКолонки = 5;
				Область.ЦветФона = WebЦвета.СветлоЗолотистый;

				Столб = Столб + 1;
				// Из регистра критерии сотрудников получить значения критериев по
сотруднику
				// Настроить в регистре сведений
				// Отобразить на табдоке эти значения
				Для Кол = 1 По МатрицаНазначений.ШиринаТаблицы Цикл
					ШиринаКолонки = 0;
					Для Стр = 1 По МатрицаНазначений.ВысотаТаблицы Цикл
						Область = МатрицаНазначений.Область("R"+
Формат(Стр,"ЧГ=0") + "C" + Формат(Кол,"ЧГ=0"));
```

```
                                        ШиринаОбласти = Область.Отступ +
СтрДлина(СокрЛП(Область.Текст));
                                        ШиринаКолонки = Макс(ШиринаКолонки,
ШиринаОбласти + 3);
                                        Если ШиринаКолонки > 40 Тогда
                                                Область.ШиринаКолонки = 40;
                                        Иначе
                                                Область.ШиринаКолонки = ШиринаКолонки;
                                        КонецЕсли;
                                Конеццикла;
                        КонецЦикла;
                        Месяц = Месяц(ТекущаяДата());
                        //
                        АктивныеЗадачи = ЗадачиАктивные(Месяц, Исполнитель.Наименование);
                        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц,
Исполнитель.Наименование);
                        ВыполненныеВСрок = ЗадачиВСрок(Месяц, Исполнитель.Наименование);
                        Если Не (ВыполненныеВСрок + СНарушениемСрока) = 0 Тогда
                                Сроки = Формат(ВыполненныеВСрок/(ВыполненныеВСрок +
СНарушениемСрока), "ЧЦ=10; ЧДЦ=3");
                                СрокиВПроцентах = Формат(Сроки/100, "ЧЦ=10; ЧДЦ=3");
                        Иначе СрокиВПроцентах = 0;
                                Если Число(СрокиВПроцентах) > 80 Тогда
                                        РасчетныеСроки = ЗначениеЦелиСроки();
                                ИначеЕсли Число(СрокиВПроцентах) > 50 Тогда
                                        РасчетныеСроки = ЗначениеНормыСроки();
                                Иначе РасчетныеСроки = ЗначениеБазыСроки();
                                КонецЕсли;
                        КонецЕсли;
                        Качество = ВернутьКачество(Исполнитель.Наименование);
                        КачествоВПроцентах = Формат(Качество/100, "ЧЦ=10; ЧДЦ=3");
                        Если КачествоВПроцентах = "" Тогда КачествоВПроцентах = 0;
                        КонецЕсли;

                        Если Число(КачествоВПроцентах) > 80 Тогда
                                РасчетноеКачество = ЗначениеЦелиКачество();
                        ИначеЕсли Число(КачествоВПроцентах) > 50 Тогда
                                РасчетноеКачество = ЗначениеНормыКачество();
                        Иначе РасчетноеКачество = ЗначениеБазыКачество();
                        КонецЕсли;

                        Количество = ВернутьКоличество(Исполнитель.Наименование);
                        КоличествоВПроцентах = Формат(Количество/100, "ЧЦ=10; ЧДЦ=3");
                        Если КоличествоВПроцентах = "" Тогда КоличествоВПроцентах = 0;
                        КонецЕсли;

                        Если Число(КоличествоВПроцентах) > 80 Тогда
                                РасчетноеКоличество = ЗначениеЦелиКоличество();
                        ИначеЕсли Число(КоличествоВПроцентах) > 50 Тогда
                                РасчетноеКоличество = ЗначениеНормыКоличество();
                        Иначе РасчетноеКоличество = ЗначениеБазыКоличество();
                        КонецЕсли;

                        ЗначениеЯчейкиДляРасчета = Формат(НормализующееЗначение1 *
НормализующееЗначение2 * (РасчетныеСроки + РасчетноеКоличество + РасчетноеКачество), "ЧЦ=10;
ЧДЦ=3");
```

```
                        ОбластьВывода = МатрицаНазначений.Область(Стрк, Столб - 1, Стрк,
Столб - 1);
                        ОбластьВывода.Текст = ЗначениеЯчейкиДляРасчета;
                        ОбластьВывода.ЦветФона = WebЦвета.Бежевый;

                    КонецЦикла;


                    // Запись задач в строки
                    Область = МатрицаНазначений.Область(Стрк, 1, Стрк, 1);
                    Область.Текст = Задача.Наименование;
                    Область.ШиринаКолонки = 5;
                    Область.ЦветФона = WebЦвета.СветлоРозовый;

                    Для Кол = 1 По МатрицаНазначений.ШиринаТаблицы Цикл
                        ШиринаКолонки = 0;
                        Для Стр = 1 По МатрицаНазначений.ВысотаТаблицы Цикл
                            Область = МатрицаНазначений.Область("R"+ Формат(Стр,"ЧГ=0")
+ "C" + Формат(Кол,"ЧГ=0"));
                            ШиринаОбласти = Область.Отступ +
СтрДлина(СокрЛП(Область.Текст));
                            ШиринаКолонки = Макс(ШиринаКолонки, ШиринаОбласти + 3);
                            Если ШиринаКолонки > 40 Тогда
                                Область.ШиринаКолонки = 40;
                            Иначе
                                Область.ШиринаКолонки = ШиринаКолонки;
                            КонецЕсли;
                        Конеццикла;
                    КонецЦикла;
                    Стрк = Стрк + 1;

            КонецЦикла;

            НомерСтр = 2;
            Пока НомерСтр <= (ЗадачиНаРаспределение.Количество() + 1) Цикл
                НомерСтолбца = 2;
                Пока НомерСтолбца <= (1 + Исполнители.Количество() *
ЗадачиНаРаспределение.Количество()) Цикл
                Область = МатрицаНазначений.Область(НомерСтр, НомерСтолбца,
НомерСтр, НомерСтолбца);
                Если Область.Текст <> "" Тогда
                    Если (НомерСтолбца + Исполнители.Количество()) <= (1 +
Исполнители.Количество() * ЗадачиНаРаспределение.Количество()) Тогда
                        ПолучитьОбласть =
МатрицаНазначений.Область(НомерСтр, НомерСтолбца + Исполнители.Количество(), НомерСтр,
НомерСтолбца + Исполнители.Количество());
                        ПолучитьОбласть.Текст = Область.Текст;
                    КонецЕсли;
                    Если (НомерСтолбца - Исполнители.Количество()) > 1 Тогда
                        ПолучитьОбласть2 =
МатрицаНазначений.Область(НомерСтр, НомерСтолбца - Исполнители.Количество(), НомерСтр,
НомерСтолбца - Исполнители.Количество());
                        ПолучитьОбласть2.Текст = Область.Текст;
                    КонецЕсли;
                КонецЕсли;
                НомерСтолбца = НомерСтолбца + 1;
            КонецЦикла;
```

```
                НомерСтр = НомерСтр + 1;
            КонецЦикла;

            НомерСтр = 2;
            Пока НомерСтр <= (ЗадачиНаРаспределение.Количество() + 1) Цикл
                НомерСтолбца = 1 + Исполнители.Количество() *
ЗадачиНаРаспределение.Количество();
                Пока НомерСтолбца > 1 Цикл
                    Область = МатрицаНазначений.Область(НомерСтр, НомерСтолбца,
НомерСтр, НомерСтолбца);
                    Если Область.Текст <> "" Тогда
                        Если (НомерСтолбца + Исполнители.Количество()) <= (1 +
Исполнители.Количество() * ЗадачиНаРаспределение.Количество()) Тогда
                            ПолучитьОбласть =
МатрицаНазначений.Область(НомерСтр, НомерСтолбца + Исполнители.Количество(), НомерСтр,
НомерСтолбца + Исполнители.Количество());
                            ПолучитьОбласть.Текст = Область.Текст;
                        КонецЕсли;
                        Если (НомерСтолбца - Исполнители.Количество()) > 1 Тогда
                            ПолучитьОбласть2 =
МатрицаНазначений.Область(НомерСтр, НомерСтолбца - Исполнители.Количество(), НомерСтр,
НомерСтолбца - Исполнители.Количество());
                            ПолучитьОбласть2.Текст = Область.Текст;
                        КонецЕсли;
                    КонецЕсли;
                    НомерСтолбца = НомерСтолбца - 1;
                КонецЦикла;
                НомерСтр = НомерСтр + 1;
            КонецЦикла;

        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница4
Тогда
            Шаг = "Шаг 4";

            // Найти минимальный элемент в строке
            // Если нет 0 в строке Вычесть минимальный из всех элементов строки
            // На стыке сумму минимальных элементов строки. Из всех этих коэф выбирается
максимальный

            Для Стр = 2 По (ЗадачиНаРаспределение.Количество() + 1) Цикл
                МинимальноеЗначениеСтр = 1000000;
                Для Столб = 2 По (Исполнители.Количество() *
ЗадачиНаРаспределение.Количество() + 1) Цикл
                    Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                    Если Область.Текст = "" Тогда ЗначениеНаСравнение = 0;
                    Иначе ЗначениеНаСравнение = Число(Область.Текст);
                    КонецЕсли;
                    Если ЗначениеНаСравнение <> 0 Тогда
                        Если Число(ЗначениеНаСравнение) < МинимальноеЗначениеСтр
Тогда
                            МинимальноеЗначениеСтр =
Число(ЗначениеНаСравнение);
                        КонецЕсли;
                    КонецЕсли;
                КонецЦикла;

                Если Не МинимальноеЗначениеСтр = 0 Тогда
```

```
                                Для Столб = 2 По (Исполнители.Количество() *
ЗадачиНаРаспределение.Количество() + 1) Цикл
                                        Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                                        Если Область.Текст = "" Тогда ЗначениеНаСравнение = 0;
                                        Иначе ЗначениеНаСравнение = Число(Область.Текст);
                                        КонецЕсли;
                                        Если ЗначениеНаСравнение <> 0 Тогда
                                                ЗначениеЯчейки = Число(ЗначениеНаСравнение) -
МинимальноеЗначениеСтр;
                                                Область.Текст = ЗначениеЯчейки;
                                        КонецЕсли;
                                КонецЦикла;
                        КонецЕсли;
                КонецЦикла;

                // Найти минимальное значение в столбце
                // Вычесть из столбца
                МинимальноеЗначениеСтолбца = Новый Массив();

                Для Столб = 2 По (Исполнители.Количество() * ЗадачиНаРаспределение.Количество() + 1)
Цикл
                        МинимальноеЗначениеСтолб = 1000000;
                        Для Стр = 2 По (ЗадачиНаРаспределение.Количество() + 1) Цикл
                                Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                                Если Область.Текст = "" Тогда ЗначениеНаСравнение = 0;
                                Иначе ЗначениеНаСравнение = Число(Область.Текст);
                                КонецЕсли;
                                Если ЗначениеНаСравнение <> 0 Тогда
                                        Если Число(ЗначениеНаСравнение) < МинимальноеЗначениеСтолб
Тогда
                                                МинимальноеЗначениеСтолб =
Число(ЗначениеНаСравнение);
                                        КонецЕсли;
                                КонецЕсли;
                        КонецЦикла;
                        МинимальноеЗначениеСтолбца.Добавить(МинимальноеЗначениеСтолб);

                        Если Не МинимальноеЗначениеСтолб = 0 Тогда
                                Для Стр = 2 По (ЗадачиНаРаспределение.Количество() + 1) Цикл
                                        Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                                        Если Область.Текст = "" Тогда ЗначениеНаСравнение = 0;
                                        Иначе ЗначениеНаСравнение = Число(Область.Текст);
                                        КонецЕсли;
                                        Если ЗначениеНаСравнение <> 0 Тогда
                                                ЗначениеЯчейки = Число(ЗначениеНаСравнение) -
МинимальноеЗначениеСтолб/2;
                                                Область.Текст = ЗначениеЯчейки;
                                        КонецЕсли;
                                КонецЦикла;
                        КонецЕсли;
                КонецЦикла;

                // Пересчитать минимумы и коэффициенты в структуру или тз, чтобы вытаскивать удобно
было
                // Новый минимум строки
                Для Стр = 2 По (ЗадачиНаРаспределение.Количество() + 1) Цикл
                        МинимальноеЗначениеСтр = 1000000;
```

```
                Для Столб = 2 По (Исполнители.Количество() *
ЗадачиНаРаспределение.Количество() + 1) Цикл
                        Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                        Если Область.Текст = "" Тогда ЗначениеНаСравнение = 0;
                        Иначе ЗначениеНаСравнение = Число(Область.Текст);
                        КонецЕсли;
                        Если ЗначениеНаСравнение <> 0 Тогда
                                Если Число(ЗначениеНаСравнение) < МинимальноеЗначениеСтр
Тогда
                                        МинимальноеЗначениеСтр =
Число(ЗначениеНаСравнение);
                                КонецЕсли;
                        КонецЕсли;

                КонецЦикла;
                НоваяСтрока = СтрМин.Добавить();
                НоваяСтрока.НомерСтроки = Стр;
                НоваяСтрока.Значение = МинимальноеЗначениеСтр;
        КонецЦикла;

        // Новый минимум столбца
        Для Столб = 2 По (Исполнители.Количество() * ЗадачиНаРаспределение.Количество() + 1)
Цикл
                МинимальноеЗначениеСтолб = 1000000;
                Для Стр = 2 По (ЗадачиНаРаспределение.Количество() + 1) Цикл
                        Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                        Если Область.Текст = "" Тогда ЗначениеНаСравнение = 0;
                        Иначе ЗначениеНаСравнение = Число(Область.Текст);
                        КонецЕсли;
                        Если ЗначениеНаСравнение <> 0 Тогда
                                Если Число(ЗначениеНаСравнение) < МинимальноеЗначениеСтолб
Тогда
                                        МинимальноеЗначениеСтолб =
Число(ЗначениеНаСравнение);
                                КонецЕсли;
                        КонецЕсли;
                КонецЦикла;
                НоваяСтрока = СтлбМин.Добавить();
                НоваяСтрока.НомерСтолбца = Столб;
                НоваяСтрока.Значение = МинимальноеЗначениеСтолб;
        КонецЦикла;

        // записать куда-то и извлекать для подсчета коэф
        Для Каждого Стр Из СтрМин Цикл
                Для Каждого Ст Из СтлбМин Цикл
                        Коэффициент =  Стр.Значение + Ст.Значение;

                        Запись = ТЗ.Добавить();
                        Запись.ЗначениеКоэффициента = Коэффициент;
                        Запись.Строка = Стр.НомерСтроки;
                        Запись.Столбец = Ст.НомерСтолбца;
                КонецЦикла;
        КонецЦикла;

        МаксимальноеКоличество =
ЗадачиНаРаспределение.Количество()/Исполнители.Количество();
```

```
                // Находим все 0 для задачи, допустим, и сравниваем коэффициенты. Максимальный - решение, удаляем
                Для Стр = 2 По (ЗадачиНаРаспределение.Количество() + 1) Цикл
                    Для Столб = 2 По (Исполнители.Количество() * ЗадачиНаРаспределение.Количество() + 1) Цикл
                        Область = МатрицаНазначений.Область(Стр, Столб, Стр, Столб);
                        Если Область.Текст = "" Тогда
                            МаксимумКоэффициента = -1;
                            ЗначениеНаСравнение = 0;
                            Для Каждого СтрКоэф Из ТЗ Цикл
                                Если СтрКоэф.Строка = Стр И СтрКоэф.Столбец = Столб Тогда
                                    ЗначениеКоэффициента = СтрКоэф.ЗначениеКоэффициента;
                                    Если ЗначениеКоэффициента > МаксимумКоэффициента Тогда
                                        МаксимумКоэффициента = ЗначениеКоэффициента;
                                        СтрокаНаЗапоминание = СтрКоэф.Строка;
                                        СтолбецНаЗапоминание = СтрКоэф.Столбец;
                                    КонецЕсли;
                                КонецЕсли;
                            КонецЦикла;
                        КонецЕсли;
                    КонецЦикла;
                    // Проверка способности взять в этот срок
                    Для Счетчик = 1 По 500 Цикл
                        ОбластьИсполнителя = МатрицаНазначений.Область(1, СтолбецНаЗапоминание, 1, СтолбецНаЗапоминание);
                        ОбластьЗадачи = МатрицаНазначений.Область(СтрокаНаЗапоминание, 1, СтрокаНаЗапоминание, 1);
                        Если ПроверкаЗадачи(ОбластьИсполнителя.Текст, ОбластьЗадачи.Текст, МаксимальноеКоличество) = Истина Тогда
                            СозданиеНаСервере(СтрокаНаЗапоминание, СтолбецНаЗапоминание);
                            Прервать;
                        Иначе
                            Для Каждого СтрТЗ Из ТЗ Цикл
                                Если СтрТЗ.Строка = СтрокаНаЗапоминание И СтрТЗ.Столбец = СтолбецНаЗапоминание Тогда
                                    СтрТЗ.ЗначениеКоэффициента = 0;
                                КонецЕсли;
                            КонецЦикла;
                            МаксимумКоэффициента = -1;
                            ЗначениеНаСравнение = 0;
                            Для Каждого СтрКоэф Из ТЗ Цикл
                                Если СтрКоэф.Строка = Стр И СтрКоэф.Столбец = Столб Тогда
                                    ЗначениеКоэффициента = СтрКоэф.ЗначениеКоэффициента;
                                    Если ЗначениеКоэффициента > МаксимумКоэффициента Тогда
                                        МаксимумКоэффициента = ЗначениеКоэффициента;
                                        СтрокаНаЗапоминание = СтрКоэф.Строка;
```

```
                                                    СтолбецНаЗапоминание =
СтрКоэф.Столбец;
                                                        КонецЕсли;
                                                КонецЕсли;
                                            КонецЦикла;
                                        КонецЕсли;
                                КонецЦикла;
                        КонецЦикла;
        КонецЕсли;
КонецПроцедуры

&НаСервере
Функция ПроверкаЗадачи(ИсполнительЗадачи, ОбластьЗадачи, МаксимальноеКоличество)
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Задачи.Исполнитель КАК Исполнитель,
        |       Задачи.Ссылка КАК Ссылка,
        |       Задачи.ДатаСдачи КАК ДатаСдачи,
        |       Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
        |       Задачи.СтатусГотовности КАК СтатусГотовности,
        |       Задачи.ДатаНачала КАК ДатаНачала,
        |       Задачи.ДатаОкончания КАК ДатаОкончания
        |ИЗ
        |       Справочник.Задачи КАК Задачи
        |ГДЕ
        |       Задачи.Исполнитель = &ИсполнительЗадачи";

        Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

        РезультатЗапроса = Запрос.Выполнить();
        Задача = Справочники.Задачи.НайтиПоНаименованию(ОбластьЗадачи);
        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
        НаВозврат = Истина;
        // Последовательное выполнение задач
        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если ВыборкаДетальныеЗаписи.ДатаНачала <= Задача.ДатаНачала И
ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения >= Задача.КрайняяДатаВнедрения
                        Тогда НаВозврат = Ложь;
                КонецЕсли;
        КонецЦикла;

        //Если ВыборкаДетальныеЗаписи.Количество() > МаксимальноеКоличество Тогда
        //      НаВозврат = Ложь;
        //КонецЕсли;

        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       ОтпускСотрудника.Сотрудник КАК Сотрудник,
        |       ОтпускСотрудника.ДатаНачала КАК ДатаНачала,
        |       ОтпускСотрудника.ДатаОкончания КАК ДатаОкончания
        |ИЗ
        |       Документ.ОтпускСотрудника КАК ОтпускСотрудника
        |ГДЕ
        |       ОтпускСотрудника.Сотрудник = &ИсполнительЗадачи";
```

```
	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

	РезультатЗапроса = Запрос.Выполнить();

	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	// Избежание попадания задачи в период отпуска сотрудника
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если ВыборкаДетальныеЗаписи.ДатаНачала <= Задача.ДатаНачала И
ВыборкаДетальныеЗаписи.ДатаОкончания >= Задача.КрайняяДатаВнедрения Тогда
			НаВозврат = Ложь;
		КонецЕсли;
	КонецЦикла;

	Возврат НаВозврат;
КонецФункции

&НаСервере
Процедура СозданиеНаСервере(СтрокаЗаписи, СтолбецЗаписи)
	// Добавление записи в регистр сведений
	НоваяЗаписьРегистра = РегистрыСведений.НазначенныеЗадачи.СоздатьМенеджерЗаписи();
	ОбластьЗадачи = МатрицаНазначений.Область(СтрокаЗаписи, 1, СтрокаЗаписи, 1);
	НоваяЗаписьРегистра.Задача = Справочники.Задачи.НайтиПоНаименованию(ОбластьЗадачи.Текст);
	ОбластьИсполнителя = МатрицаНазначений.Область(1, СтолбецЗаписи, 1, СтолбецЗаписи);
	НоваяЗаписьРегистра.Исполнитель =
Справочники.Исполнители.НайтиПоНаименованию(ОбластьИсполнителя.Текст);
	ОбъектЗадачи = Справочники.Задачи.НайтиПоНаименованию(ОбластьЗадачи.Текст);
	НоваяЗаписьРегистра.ДатаСдачи = ОбъектЗадачи.КрайняяДатаВнедрения;
	НоваяЗаписьРегистра.Записать();

	ЗадачаПолучение = ОбъектЗадачи.ПолучитьОбъект();
	ЗадачаПолучение.Исполнитель =
Справочники.Исполнители.НайтиПоНаименованию(ОбластьИсполнителя.Текст);
	ЗадачаПолучение.Записать();
КонецПроцедуры

&НаСервере
Процедура ПолучитьСписокИсполнителей()
	// Получение списка исполнителей
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Исполнители.Наименование КАК Наименование,
	|	Исполнители.ОбластьСпециализации КАК ОбластьСпециализации,
	|	Исполнители.Загруженность КАК Загруженность,
	|	Исполнители.КритерийЭффективности КАК КритерийЭффективности,
	|	Исполнители.Отпуск.(
	|		ДатаНачала КАК ДатаНачала,
	|		ДатаОкончания КАК ДатаОкончания
	|	) КАК Отпуск
	|ИЗ
	|	Справочник.Исполнители КАК Исполнители";

	РезультатЗапроса = Запрос.Выполнить().Выгрузить();
	Исполнители.Загрузить(РезультатЗапроса);
КонецПроцедуры
```

```
&НаСервере
Процедура ПолучитьСписокЗадач()
	// Получение списка задач
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.Приоритет КАК Приоритет,
	|	Задачи.УровеньСложности КАК УровеньСложности,
	|	Задачи.ТипЗадачи КАК ТипЗадачи,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаНачала КАК ДатаНачала,
	|	Задачи.Исполнитель КАК Исполнитель
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.ДатаНачала >= &ДатаНачала
	|	И Задачи.Исполнитель = &Исполнитель";
	Запрос.УстановитьПараметр("ДатаНачала", ТекущаяДата());
	Запрос.УстановитьПараметр("Исполнитель", Справочники.Исполнители.ПустаяСсылка());
	РезультатЗапроса = Запрос.Выполнить().Выгрузить();
	ЗадачиНаРаспределение.Загрузить(РезультатЗапроса);
КонецПроцедуры

&НаСервере
Функция РасчетПервогоПараметра(ОбластьСпециализации, ТипЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	ТипыЗадач.Ссылка КАК Ссылка
	|ИЗ
	|	Справочник.ТипыЗадач КАК ТипыЗадач";

	РезультатЗапроса = Запрос.Выполнить();
	ОбщееКоличество = РезультатЗапроса.Выбрать().Количество();

	// Если Область специализации совпадает с типом задачи, нормализующее значение
устанавливается в максимум согласно ограничениям в пояснительной записке
	Если ОбластьСпециализации = ТипЗадачи Тогда
		НормализующееЗначение1 = 5;
	Иначе
		Запрос = Новый Запрос;
		Запрос.Текст =
		"ВЫБРАТЬ
		|	СоответствиеТиповЗадач.Тип КАК Тип,
		|	СоответствиеТиповЗадач.СмежныйТип КАК СмежныйТип
		|ИЗ
		|	Справочник.СоответствиеТиповЗадач КАК СоответствиеТиповЗадач";

		РезультатЗапроса = Запрос.Выполнить();

		ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
		Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

			// Если область специализации не совпадает с типом задачи,
```

```
                    // Осуществляется поиск смежного типа и нормализующее значение
устанавливается как процент от максимального значения
                    Если ОбластьСпециализации = ВыборкаДетальныеЗаписи.Тип И ТипЗадачи =
ВыборкаДетальныеЗаписи.СмежныйТип ИЛИ
                        ОбластьСпециализации = ВыборкаДетальныеЗаписи.СмежныйТип И
ТипЗадачи = ВыборкаДетальныеЗаписи.Тип Тогда
                        НормализующееЗначение1 = (ОбщееКоличество - 1)/ОбщееКоличество *
5;

                        // Если тип задачи не является смежным типом, находится смежный
смежного и нормализующее значение принимает меньшее значение
                    ИначеЕсли ОбластьСпециализации = ВыборкаДетальныеЗаписи.Тип Тогда
                        ЗапросВнутренний = Новый Запрос;
                        ЗапросВнутренний.Текст =
                        "ВЫБРАТЬ
                        |       СоответствиеТиповЗадач.Тип КАК Тип,
                        |       СоответствиеТиповЗадач.СмежныйТип КАК СмежныйТип
                        |ИЗ
                        |       Справочник.СоответствиеТиповЗадач КАК
СоответствиеТиповЗадач";

                        РезультатЗапроса = ЗапросВнутренний.Выполнить();

                        ВнутренняяВыборка = РезультатЗапроса.Выбрать();

                        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                            Если ВнутренняяВыборка.Тип =
ВыборкаДетальныеЗаписи.СмежныйТип И ВнутренняяВыборка.СмежныйТип = ТипЗадачи ИЛИ
                                ВнутренняяВыборка.Тип = ТипЗадачи И
ВнутренняяВыборка.СмежныйТип = ВыборкаДетальныеЗаписи.СмежныйТип Тогда
                                НормализующееЗначение1 = (ОбщееКоличество -
2)/ОбщееКоличество * 5;
                            КонецЕсли;
                        КонецЦикла;
                    Иначе НормализующееЗначение1 = (ОбщееКоличество - 3)/ОбщееКоличество * 5;
                    КонецЕсли;

                КонецЦикла;
        КонецЕсли;
        Возврат НормализующееЗначение1;
КонецФункции

&НаСервере
Функция РасчетВторогоПараметра(Исполнитель, ОбластьСпециализации, ТипЗадачи)
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Сертификаты.ИмяДокумента КАК ИмяДокумента,
        |       Сертификаты.Область КАК Тип,
        |       Сертификаты.Сотрудник КАК Сотрудник
        |ИЗ
        |       Документ.Сертификат КАК Сертификаты
        |ГДЕ
        |       Сертификаты.Сотрудник = &Сотрудник";

        Запрос.УстановитьПараметр("Сотрудник",
Справочники.Исполнители.НайтиПоНаименованию(Исполнитель));
```

```
	РезультатЗапроса = Запрос.Выполнить();

	НормализующееЗначение2 = 10;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если ВыборкаДетальныеЗаписи.Тип = ОбластьСпециализации Тогда
			НормализующееЗначение2 = 10;
		ИначеЕсли ТипЗадачи = Перечисления.ТипыЗадач.Сопровождение Тогда
			Если ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Внедрение Тогда
				НормализующееЗначение2 = 8;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.УстановкаПО Тогда
				НормализующееЗначение2 = 5;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.ПодключениеСервисов Тогда
				НормализующееЗначение2 = 2;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Прочие
Тогда
				НормализующееЗначение2 = 1;
			КонецЕсли;
		ИначеЕсли ТипЗадачи = Перечисления.ТипыЗадач.Внедрение Тогда
			Если ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Сопровождение
Тогда
				НормализующееЗначение2 = 8;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.УстановкаПО Тогда
				НормализующееЗначение2 = 5;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.ПодключениеСервисов Тогда
				НормализующееЗначение2 = 2;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Прочие
Тогда
				НормализующееЗначение2 = 1;
			КонецЕсли;
		ИначеЕсли ТипЗадачи = Перечисления.ТипыЗадач.УстановкаПО Тогда
			Если ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Внедрение Тогда
				НормализующееЗначение2 = 5;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.Сопровождение Тогда
				НормализующееЗначение2 = 5;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.ПодключениеСервисов Тогда
				НормализующееЗначение2 = 8;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Прочие
Тогда
				НормализующееЗначение2 = 1;
			КонецЕсли;
		ИначеЕсли ТипЗадачи = Перечисления.ТипыЗадач.ПодключениеСервисов Тогда
			Если ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Внедрение Тогда
				НормализующееЗначение2 = 4;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.Сопровождение Тогда
				НормализующееЗначение2 = 4;
			ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.УстановкаПО Тогда
```

```
                        НормализующееЗначение2 = 7;
                    ИначеЕсли ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Прочие
Тогда
                        НормализующееЗначение2 = 1;
                    КонецЕсли;
                ИначеЕсли ТипЗадачи = Перечисления.ТипыЗадач.Прочие Тогда
                    Если ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Внедрение Тогда
                        НормализующееЗначение2 = 1;
                    ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.Сопровождение Тогда
                        НормализующееЗначение2 = 1;
                    ИначеЕсли ВыборкаДетальныеЗаписи.Тип =
Перечисления.ТипыЗадач.ПодключениеСервисов Тогда
                        НормализующееЗначение2 = 1;
                    ИначеЕсли ВыборкаДетальныеЗаписи.Тип = Перечисления.ТипыЗадач.Прочие
Тогда
                        НормализующееЗначение2 = 1;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат НормализующееЗначение2;
КонецФункции

&НаКлиенте
Процедура Предыдущий(Команда)
        Если ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница1
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница1;
                Шаг = "Шаг 1";
        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница2
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница1;
                Шаг = "Шаг 1";
        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница3
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница2;
                Шаг = "Шаг 2";
        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница4
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница3;

                Шаг = "Шаг 3";
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура Следующий(Команда)
        Если ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница1
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница2;
                Шаг = "Шаг 2";
        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница2
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница3;
                Шаг = "Шаг 3";
        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница3
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница4;

                Шаг = "Шаг 4";
        ИначеЕсли ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница4
                Тогда ЭтаФорма.Элементы.Страницы.ТекущаяСтраница = ЭтаФорма.Элементы.Страница4;

                Шаг = "Шаг 4";
```

```
	КонецЕсли;
КонецПроцедуры

&НаСервере
Функция ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи
	|	И Задачи.ДатаСдачи > Задачи.КрайняяДатаВнедрения";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

	РезультатЗапроса = Запрос.Выполнить();
	КоличествоСЗадержкой = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
			Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
				КоличествоСЗадержкой = КоличествоСЗадержкой + 1;
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;

	Возврат КоличествоСЗадержкой;
КонецФункции

&НаСервере
Функция ЗадачиАктивные(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

	РезультатЗапроса = Запрос.Выполнить();
	КоличествоАктивных = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если Строка(ВыборкаДетальныеЗаписи.ДатаСдачи) = "01.01.0001 0:00:00" Тогда
			КоличествоАктивных = КоличествоАктивных + 1;
```

```
                КонецЕсли;
        КонецЦикла;

        Возврат КоличествоАктивных;
КонецФункции

&НаСервере
Функция ЗадачиВСрок(Месяц, ИсполнительЗадачи)
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Задачи.Наименование КАК Наименование,
        |       Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
        |       Задачи.ДатаСдачи КАК ДатаСдачи
        |ИЗ
        |       Справочник.Задачи КАК Задачи
        |ГДЕ
        |       Задачи.Исполнитель = &ИсполнительЗадачи
        |       И Задачи.ДатаСдачи <= Задачи.КрайняяДатаВнедрения";

        Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

        РезультатЗапроса = Запрос.Выполнить();

        КоличествоВыполненныхВСрок = 0;
        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
                        Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
                                КоличествоВыполненныхВСрок = КоличествоВыполненныхВСрок + 1;
                        КонецЕсли;
                КонецЕсли;
        КонецЦикла;

        Возврат КоличествоВыполненныхВСрок;
КонецФункции

&НаСервере
Функция ЗначениеЦелиСроки()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
```

```
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Сроки") Тогда
                    Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.Цель Тогда
                        СрокиЦель = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат СрокиЦель;
КонецФункции

&НаСервере
Функция ЗначениеНормыСроки()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Сроки") Тогда
                    Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.Норма Тогда
                        СрокиНорма = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат СрокиНорма;
КонецФункции

&НаСервере
Функция ЗначениеБазыСроки()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
```

```
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Сроки") Тогда
			Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.База Тогда
				СрокиБаза = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;
	Возврат СрокиБаза;
КонецФункции

&НаСервере
Функция ВернутьКачество(Исполнитель)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.ПредыдущийИсполнитель КАК ПредыдущийИсполнитель
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(Исполнитель));

	РезультатЗапроса = Запрос.Выполнить();
	ПоказательКачества = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если ВыборкаДетальныеЗаписи.ПредыдущийИсполнитель =
ВыборкаДетальныеЗаписи.ПредыдущийИсполнитель Тогда
			ПоказательКачества = ПоказательКачества + 1;
		КонецЕсли;
	КонецЦикла;
	Возврат ПоказательКачества;
КонецФункции

&НаСервере
Функция ЗначениеЦелиКачество()
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	КритерииЭффективности.Критерий КАК Критерий,
	|	КритерииЭффективности.Значение КАК Значение,
	|	КритерииЭффективности.Вес КАК Вес,
	|	КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
	|ИЗ
	|	РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

	РезультатЗапроса = Запрос.Выполнить();

	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
```

```
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Качество") Тогда
                    Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.Цель Тогда
                        КачествоЦель = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат КачествоЦель;

КонецФункции

&НаСервере
Функция ЗначениеНормыКачество()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Качество") Тогда
                    Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.Норма Тогда
                        КачествоНорма = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат КачествоНорма;
КонецФункции

&НаСервере
Функция ЗначениеБазыКачество()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
```

```
        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Качество") Тогда
                        Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.База Тогда
                                КачествоБаза = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                        КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат КачествоБаза;
КонецФункции

&НаСервере
Функция ВернутьКоличество(Исполнитель)
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Задачи.Наименование КАК Наименование,
        |       Задачи.ПредыдущийИсполнитель КАК ПредыдущийИсполнитель
        |ИЗ
        |       Справочник.Задачи КАК Задачи
        |ГДЕ
        |       Задачи.Исполнитель = &ИсполнительЗадачи";

        Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(Исполнитель));

        РезультатЗапроса = Запрос.Выполнить();
        ПоказательКоличества = 0;
        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                ПоказательКоличества = ПоказательКоличества + 1;
        КонецЦикла;
        Возврат ПоказательКоличества;
КонецФункции

&НаСервере
Функция ЗначениеЦелиКоличество()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Количество") Тогда
```

```
                    Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.Цель Тогда
                        КоличествоБаза = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат КоличествоБаза;
КонецФункции

&НаСервере
Функция ЗначениеНормыКоличество()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Количество") Тогда
                    Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.Норма Тогда
                        КоличествоНорма = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                    КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат КоличествоНорма;
КонецФункции

&НаСервере
Функция ЗначениеБазыКоличество()
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       КритерииЭффективности.Критерий КАК Критерий,
        |       КритерииЭффективности.Значение КАК Значение,
        |       КритерииЭффективности.Вес КАК Вес,
        |       КритерииЭффективности.УровеньЭффективности КАК УровеньЭффективности
        |ИЗ
        |       РегистрСведений.КритерииЭффективности КАК КритерииЭффективности";

        РезультатЗапроса = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
```

```
                    Если ВыборкаДетальныеЗаписи.Критерий =
Справочники.КритерииЭффективности.НайтиПоНаименованию("Количество") Тогда
                        Если ВыборкаДетальныеЗаписи.УровеньЭффективности =
Перечисления.УровниЭффективности.База Тогда
                            КоличествоБаза = ВыборкаДетальныеЗаписи.Значение *
ВыборкаДетальныеЗаписи.Вес;
                        КонецЕсли;
                    КонецЕсли;
        КонецЦикла;
        Возврат КоличествоБаза;
КонецФункции

&НаКлиенте
Процедура ПриОткрытии(Отказ)
        Шаг = "Шаг 1";
        ПолучитьСписокЗадач();
КонецПроцедуры

// Изменения статуса задачи
&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
        // Получение элемента Справочника "Задачи" для изменения статуса
        Элемент = Справочники.Задачи.НайтиПоНаименованию(Объект.Задача);
        Если Не Элемент.Пустая() Тогда
                Задача = Элемент.ПолучитьОбъект();
                Если Объект.Статус = Перечисления.СтатусыГотовностиЗадачи.ВИсполнении Тогда
                        Задача.СтатусГотовности =
Перечисления.СтатусыГотовностиЗадачи.ВИсполнении;
                ИначеЕсли Объект.Статус = Перечисления.СтатусыГотовностиЗадачи.ГотоваНоНеСдана
Тогда
                        Задача.СтатусГотовности =
Перечисления.СтатусыГотовностиЗадачи.ГотоваНоНеСдана;
                ИначеЕсли Объект.Статус = Перечисления.СтатусыГотовностиЗадачи.НеГотова Тогда
                        Задача.СтатусГотовности = Перечисления.СтатусыГотовностиЗадачи.НеГотова;
                КонецЕсли;
                Задача.Записать();
        КонецЕсли;
КонецПроцедуры

&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
        Задача = Справочники.Задачи.НайтиПоНаименованию(Объект.Задача);
        Если Не Задача.Пустая() Тогда
                Элемент = Задача.ПолучитьОбъект();
                Элемент.СтатусГотовности = Перечисления.СтатусыГотовностиЗадачи.Сдана;
                Элемент.ДатаСдачи = Объект.ДатаСдачи;
                Элемент.Записать();
        КонецЕсли;
КонецПроцедуры

// Формирование диаграмм загруженности сотрудников
&НаКлиенте
Процедура СотрудникиПриАктивизацииСтроки(Элемент)
        ИсполнительЗадачи = Элемент.ТекущиеДанные.Наименование;
        Динамика.Очистить();

        СерияАктивныеЗадачи = Динамика.Серии.Добавить("Активные задачи");
```

```
        СерияНарушениеСрокаДинамика = Динамика.Серии.Добавить("С нарушением срока");
        СерияВСрок = Загрузка.Серии.Добавить("В срок");
        СерияНарушениеСрока = Загрузка.Серии.Добавить("С нарушением срока");

        Месяц = 1;
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);

        ВыводЗагрузки(ИсполнительЗадачи, СерияВСрок, СерияНарушениеСрока, СНарушениемСрока,
ВыполненныеВСрок);

        Если Режим = "Год" Тогда
                Для Месяц = 1 По 12 Цикл
                        Если Месяц = 1 Тогда
                                АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                                СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                                ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                                ВыводДинамики("Январь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                        ИначеЕсли Месяц = 2 Тогда
                                АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                                СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                                ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                                ВыводДинамики("Февраль", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                        ИначеЕсли Месяц = 3 Тогда
                                АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                                СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                                ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                                ВыводДинамики("Март", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                        ИначеЕсли Месяц = 4 Тогда
                                АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                                СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                                ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                                ВыводДинамики("Апрель", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                        ИначеЕсли Месяц = 5 Тогда
                                АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                                СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
```

```
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                    ВыводДинамики("Май", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли Месяц = 6 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                    СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                    ВыводДинамики("Июнь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли Месяц = 7 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                    СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                    ВыводДинамики("Июль", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли Месяц = 8 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                    СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                    ВыводДинамики("Август", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли Месяц = 9 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                    СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                    ВыводДинамики("Сентябрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли Месяц = 10 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                    СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                    ВыводДинамики("Октябрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);
```

```
                ИначеЕсли Месяц = 11 Тогда
                        АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                        СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                        ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                        ВыводДинамики("Ноябрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);


                ИначеЕсли Месяц = 12 Тогда
                        АктивныеЗадачи = ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи);
                        СНарушениемСрока = ЗадачиСНарушениемСрокаДинамика(Месяц,
ИсполнительЗадачи);
                        ВыполненныеВСрок = ЗадачиВСрокДинамика(Месяц,
ИсполнительЗадачи);

                        ВыводДинамики("Декабрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                КонецЕсли;
        КонецЦикла;
    Иначе
        ДатаНачала = Месяц(Период.ДатаНачала);
        ДатаОкончания = Месяц(Период.ДатаОкончания);
        СледующийМесяц = ДатаНачала;
        Пока ДатаОкончания > СледующийМесяц Цикл
            Если СледующийМесяц = 1 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                ВыводДинамики("Январь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 2 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                ВыводДинамики("Февраль", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 3 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);
```

```
                ВыводДинамики("Март", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 4 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                ВыводДинамики("Апрель", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 5 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                ВыводДинамики("Май", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 6 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                ВыводДинамики("Июнь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 7 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                ВыводДинамики("Июль", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

            ИначеЕсли СледующийМесяц = 8 Тогда
                АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);
```

```
                    ВыводДинамики("Август", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли СледующийМесяц = 9 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                    СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                    ВыводДинамики("Сентябрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли СледующийМесяц = 10 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                    СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                    ВыводДинамики("Октябрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли СледующийМесяц = 11 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                    СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                    ВыводДинамики("Ноябрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                ИначеЕсли СледующийМесяц = 12 Тогда
                    АктивныеЗадачи = ЗадачиАктивныеДинамика(СледующийМесяц,
ИсполнительЗадачи);
                    СНарушениемСрока =
ЗадачиСНарушениемСрокаДинамика(СледующийМесяц, ИсполнительЗадачи);
                    ВыполненныеВСрок = ЗадачиВСрокДинамика(СледующийМесяц,
ИсполнительЗадачи);

                    ВыводДинамики("Декабрь", СерияАктивныеЗадачи,
СерияНарушениеСрокаДинамика, АктивныеЗадачи, СНарушениемСрока);

                КонецЕсли;

                СледующийМесяц = СледующийМесяц + 1;
            КонецЦикла;

        КонецЕсли;
КонецПроцедуры

&НаКлиенте
```

```
Процедура ВыводДинамики(Месяц, СерияАктивныеЗадачи, СерияНарушениеСрока, АктивныеЗадачи,
СНарушениемСрока)
	Точка = Динамика.УстановитьТочку(Месяц);
	Динамика.УстановитьЗначение(Точка, СерияАктивныеЗадачи, АктивныеЗадачи);
	Динамика.УстановитьЗначение(Точка, СерияНарушениеСрока, СНарушениемСрока);
КонецПроцедуры

&НаКлиенте
Процедура ВыводЗагрузки(ИсполнительЗадачи, СерияВСрок, СерияНарушениеСрока, СНарушениемСрока,
ВыполненныеВСрок)
	ТочкаЗагрузка = Загрузка.УстановитьТочку(ИсполнительЗадачи);
	Загрузка.УстановитьЗначение(ТочкаЗагрузка, СерияВСрок, ВыполненныеВСрок);
	Загрузка.УстановитьЗначение(ТочкаЗагрузка, СерияНарушениеСрока, СНарушениемСрока);
КонецПроцедуры

Функция ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи
	|	И Задачи.ДатаСдачи > Задачи.КрайняяДатаВнедрения";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));
	РезультатЗапроса = Запрос.Выполнить();
	КоличествоСЗадержкой = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		//Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
		//		Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
		Если ВыборкаДетальныеЗаписи.ДатаСдачи >
ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения Тогда
		КоличествоСЗадержкой = КоличествоСЗадержкой + 1;
		КонецЕсли;
		//		КонецЕсли;
		//КонецЕсли;
	КонецЦикла;
	Возврат КоличествоСЗадержкой;
КонецФункции

&НаСервере
Функция ЗадачиАктивные(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
```

```
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

	РезультатЗапроса = Запрос.Выполнить();
	КоличествоАктивных = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если Строка(ВыборкаДетальныеЗаписи.ДатаСдачи) = "01.01.0001 0:00:00" Тогда
			КоличествоАктивных = КоличествоАктивных + 1;
		КонецЕсли;
	КонецЦикла;
	Возврат КоличествоАктивных;
КонецФункции

&НаСервере
Функция ЗадачиВСрок(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи
	|	И Задачи.ДатаСдачи <= Задачи.КрайняяДатаВнедрения";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));
	РезультатЗапроса = Запрос.Выполнить();

	КоличествоВыполненныхВСрок = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		//Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
			//Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
				КоличествоВыполненныхВСрок = КоличествоВыполненныхВСрок + 1;
			//КонецЕсли;
		//КонецЕсли;
	КонецЦикла;

	Возврат КоличествоВыполненныхВСрок;
КонецФункции

&НаКлиенте
Процедура ПриОткрытии(Отказ)
	Режим = "Год";
	Динамика.Очистить();
	Загрузка.Очистить();
КонецПроцедуры

Функция ЗадачиСНарушениемСрокаДинамика(Месяц, ИсполнительЗадачи)
```

```
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи
	|	И Задачи.ДатаСдачи > Задачи.КрайняяДатаВнедрения";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

	РезультатЗапроса = Запрос.Выполнить();
	КоличествоСЗадержкой = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
		Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
		Если ВыборкаДетальныеЗаписи.ДатаСдачи >
ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения Тогда
		КоличествоСЗадержкой = КоличествоСЗадержкой + 1;
		КонецЕсли;
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;
	Возврат КоличествоСЗадержкой;
КонецФункции

&НаСервере
Функция ЗадачиАктивныеДинамика(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));

	РезультатЗапроса = Запрос.Выполнить();
	КоличествоАктивных = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если Строка(ВыборкаДетальныеЗаписи.ДатаСдачи) = "01.01.0001 0:00:00" Тогда
			КоличествоАктивных = КоличествоАктивных + 1;
		КонецЕсли;
	КонецЦикла;
	Возврат КоличествоАктивных;
```

```
КонецФункции

&НаСервере
Функция ЗадачиВСрокДинамика(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи
	|	И Задачи.ДатаСдачи <= Задачи.КрайняяДатаВнедрения";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));
	РезультатЗапроса = Запрос.Выполнить();

	КоличествоВыполненныхВСрок = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
			Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
				КоличествоВыполненныхВСрок = КоличествоВыполненныхВСрок + 1;
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;
	Возврат КоличествоВыполненныхВСрок;
КонецФункции

// Формирование диаграмм эффективности сотрудников
&НаКлиенте
Процедура СотрудникиПриАктивизацииСтроки(Элемент)
	Шапка = "Динамика исполнительской дисциплины: " + Элемент.ТекущиеДанные.Наименование;
	ИсполнительЗадачи = Элемент.ТекущиеДанные.Наименование;
	Серия = ИсполнительскаяДисциплина.Серии.Добавить(Элемент.ТекущиеДанные.Наименование);

	Месяц = 1;
	Пока Месяц <> 12 Цикл
	Если Месяц = 1 Тогда
		СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
		ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
		Точка = ИсполнительскаяДисциплина.УстановитьТочку("Январь");
		ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);
		//ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, СНарушениемСрока);

	ИначеЕсли Месяц = 2 Тогда
		СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
		ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
		Точка = ИсполнительскаяДисциплина.УстановитьТочку("Февраль");
		ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);
		//ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, СНарушениемСрока);

	ИначеЕсли Месяц = 3 Тогда
```

```
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Март");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);
        //ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, СНарушениемСрока);

ИначеЕсли Месяц = 4 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Апрель");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);
        //ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, СНарушениемСрока);

ИначеЕсли Месяц = 5 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Май");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);
        //ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, СНарушениемСрока);

ИначеЕсли Месяц = 6 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Июнь");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

ИначеЕсли Месяц = 7 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Июль");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

ИначеЕсли Месяц = 8 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Август");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

ИначеЕсли Месяц = 9 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Сентябрь");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

ИначеЕсли Месяц = 10 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Октябрь");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

ИначеЕсли Месяц = 11 Тогда
        СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
        ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
        Точка = ИсполнительскаяДисциплина.УстановитьТочку("Ноябрь");
        ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

ИначеЕсли Месяц = 12 Тогда
```

```
                СНарушениемСрока = ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи);
                ВыполненныеВСрок = ЗадачиВСрок(Месяц, ИсполнительЗадачи);
                Точка = ИсполнительскаяДисциплина.УстановитьТочку("Декабрь");
                ИсполнительскаяДисциплина.УстановитьЗначение(Точка, Серия, ВыполненныеВСрок);

        КонецЕсли;
        Месяц = Месяц + 1;
        КонецЦикла;
КонецПроцедуры

Функция ЗадачиСНарушениемСрока(Месяц, ИсполнительЗадачи)
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Задачи.Наименование КАК Наименование,
        |       Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
        |       Задачи.ДатаСдачи КАК ДатаСдачи
        |ИЗ
        |       Справочник.Задачи КАК Задачи
        |ГДЕ
        |       Задачи.Исполнитель = &Исполнитель
        |       И Задачи.ДатаСдачи > Задачи.КрайняяДатаВнедрения";

        Запрос.УстановитьПараметр("Исполнитель",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));
        РезультатЗапроса = Запрос.Выполнить();
        КоличествоСЗадержкой = 0;
        ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
                        Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
                                Если ВыборкаДетальныеЗаписи.ДатаСдачи >
ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения Тогда
                                КоличествоСЗадержкой = КоличествоСЗадержкой + 1;
                                КонецЕсли;
                        КонецЕсли;
                КонецЕсли;
        КонецЦикла;
        Возврат КоличествоСЗадержкой;
КонецФункции

&НаСервере
Функция ЗадачиАктивные(Месяц, ИсполнительЗадачи)
        Запрос = Новый Запрос;
        Запрос.Текст =
        "ВЫБРАТЬ
        |       Задачи.Наименование КАК Наименование,
        |       Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
        |       Задачи.ДатаСдачи КАК ДатаСдачи
        |ИЗ
        |       Справочник.Задачи КАК Задачи
        |ГДЕ
        |       Задачи.Исполнитель = &Исполнитель";

        Запрос.УстановитьПараметр("Исполнитель",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));
```

```
	РезультатЗапроса = Запрос.Выполнить();
	КоличествоАктивных = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если Строка(ВыборкаДетальныеЗаписи.ДатаСдачи) = "01.01.0001 0:00:00" Тогда
			КоличествоАктивных = КоличествоАктивных + 1;
		КонецЕсли;
	КонецЦикла;
	Возврат КоличествоАктивных;
КонецФункции

&НаСервере
Функция ЗадачиВСрок(Месяц, ИсполнительЗадачи)
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Задачи.Наименование КАК Наименование,
	|	Задачи.КрайняяДатаВнедрения КАК КрайняяДатаВнедрения,
	|	Задачи.ДатаСдачи КАК ДатаСдачи
	|ИЗ
	|	Справочник.Задачи КАК Задачи
	|ГДЕ
	|	Задачи.Исполнитель = &ИсполнительЗадачи
	|	И Задачи.ДатаСдачи <= Задачи.КрайняяДатаВнедрения";

	Запрос.УстановитьПараметр("ИсполнительЗадачи",
Справочники.Исполнители.НайтиПоНаименованию(ИсполнительЗадачи));
	РезультатЗапроса = Запрос.Выполнить();

	КоличествоВыполненныхВСрок = 0;
	ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		Если СтрДлина(Месяц) = 1 Тогда МесяцДляПроверки = "0" + Строка(Месяц);
			Если МесяцДляПроверки =
Сред(Строка(ВыборкаДетальныеЗаписи.КрайняяДатаВнедрения), 4, 2) Тогда
				КоличествоВыполненныхВСрок = КоличествоВыполненныхВСрок + 1;
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;
	Возврат КоличествоВыполненныхВСрок;
КонецФункции

&НаКлиенте
Процедура ПриОткрытии(Отказ)
	ИсполнительскаяДисциплина.Очистить();
КонецПроцедуры

// Создание нового пользователя программным способом при создании пользователя в
// пользовательском режиме в справочнике «Пользователи»
&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
	Пользователь = ПользователиИнформационнойБазы.НайтиПоИмени(Объект.Наименование);
	РольПользователя = Метаданные.Роли.Найти(Объект.Роль);
	Если Пользователь = Неопределено Тогда
		Пользователь = ПользователиИнформационнойБазы.СоздатьПользователя();
		Пользователь.Имя = Объект.Наименование;
```

```
                Пользователь.ПолноеИмя = Объект.Наименование;
                Пользователь.Пароль = Объект.Пароль;
                Пользователь.Роли.Добавить(РольПользователя);
                Пользователь.Записать();
        ИначеЕсли Пользователь.Имя = Объект.Наименование Тогда // Если пользователь уже записан
                Пользователь.Пароль = Объект.Пароль;
                Если Пользователь.Роли.Содержит(РольПользователя) Тогда // Роль определена верно =>
Записывается
                        Пользователь.Записать();
                Иначе // Роль не та, что мы задаем => Меняем
                Пользователь.Роли.Очистить();
                Пользователь.Роли.Добавить(РольПользователя);
                Пользователь.Записать();
                КонецЕсли;
        КонецЕсли;
КонецПроцедуры

// Проверка введенного пароля с помощью регилярных выражений
// Пароль должен содержать прописные и заглавные буквы латинского алфавита в
// количестве более 1 шт. для каждого, не менее 1 цифры, не менее 1 спецсимвола.
&НаСервере
Процедура ПроверитьПарольНаСервере()
        ШаблонПароля = "(?=.*[0-9])(?=.*[!@#$%^&;*])(?=.*[a-z])(?=.*[A-Z])[0-9a-zA-Z!@#$%^&;*]{6,}";

        RegExp = Новый COMОбъект("VBScript.RegExp");
        RegExp.MultiLine = Ложь;
        RegExp.Global = Истина;
        RegExp.IgnoreCase = Ложь;
        RegExp.Pattern = ШаблонПароля;
        Проверка = Ложь;
        Если RegExp.Test(Объект.Пароль) Тогда
                Сообщить("Корректный пароль");
                Проверка = Истина;
        Иначе
                Сообщить("Некорректный пароль");
                Проверка = Ложь;
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
        // Отказ от записи объекта при несоответствии пароля шаблону
        ПроверитьПарольНаСервере();
        Если Проверка = Ложь Тогда
                Отказ = Истина;
        КонецЕсли;
КонецПроцедуры

// Обмен данными на стороне мобильного клиента
&НаСервере
Процедура ОбменНаСервере()
        Определения = Новый WSОпределения(Адрес);
        URI = "http://localhost/ws1";
        Прокси = Новый WSПрокси(Определения, URI, "WebОбмен", "WebОбменSoap");

        СисИнфо = Новый СистемнаяИнформация;
        КодМобильногоКомпьютера = Строка(СисИнфо.ИдентификаторКлиента);
```

```
	Узел = ПланыОбмена.Мобильные.ЭтотУзел();
	Если Не ЗначениеЗаполнено(Узел.Код) ИЛИ Узел.Код<>КодМобильногоКомпьютера Тогда
		УзелОбмена = Узел.ПолучитьОбъект();
		УзелОбмена.Код = КодМобильногоКомпьютера;
		УзелОбмена.Наименование = КодМобильногоКомпьютера;
		УзелОбмена.Записать();
	КонецЕсли;

	ЦентральныйУзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду("001");
	Если ЦентральныйУзелОбмена.Пустая() Тогда
		НовыйУзел = ПланыОбмена.Мобильные.СоздатьУзел();
		НовыйУзел.Код = "001";
		НовыйУзел.Наименование = "Центральный";
		НовыйУзел.Записать();
		ЦентральныйУзелОбмена = НовыйУзел.Ссылка;
	КонецЕсли;

	ДанныеОбмена = Прокси.ВыполнитьОбмен(КодМобильногоКомпьютера,
СформироватьПакетОбмена(ЦентральныйУзелОбмена));

	ЧтениеXML = Новый ЧтениеXML;
	ЧтениеXML.УстановитьСтроку(ДанныеОбмена.Получить());
	ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
	ЧтениеСообщения.НачатьЧтение(ЧтениеXML);
	ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
ЧтениеСообщения.НомерПринятого);

	НачатьТранзакцию();

	Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
		Данные = ПрочитатьXML(ЧтениеXML);
		Если Не Данные = Неопределено Тогда
			Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
			Данные.ОбменДанными.Загрузка = Истина;
			Данные.Записать();
		КонецЕсли;
	КонецЦикла;

	ЗафиксироватьТранзакцию();
	ЧтениеСообщения.ЗакончитьЧтение();
	ЧтениеXML.Закрыть();
КонецПроцедуры

Функция СформироватьПакетОбмена(УзелОбмена) Экспорт
	ЗаписьXML = Новый ЗаписьXML;
	ЗаписьXML.УстановитьСтроку("UTF-8");
	ЗаписьXML.ЗаписатьОбъявлениеXML();
	ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
	ЗаписьСообщения.НачатьЗапись(ЗаписьXML,ПланыОбмена.Мобильные.НайтиПоКоду("001"));

	ЗаписьXML.ЗаписатьСоответствиеПространстваИмен("xsi", "http://www.w3.org/2001/XMLSchema-
instanse");
	ЗаписьXML.ЗаписатьСоответствиеПространстваИмен("v8", "http://v8.1c.ru/data");
	ВыборкаИзменений = ПланыОбмена.ВыбратьИзменения(УзелОбмена,
ЗаписьСообщения.НомерСообщения);
```

```
        Пока ВыборкаИзменений.Следующий() Цикл
                Данные = ВыборкаИзменений.Получить();
                ЗаписатьXML(ЗаписьXML, Данные);
        КонецЦикла;

        ЗаписьСообщения.ЗакончитьЗапись();
        Возврат Новый ХранилищеЗначения(ЗаписьXML.Закрыть(), новый СжатиеДанных());
КонецФункции

&НаКлиенте
Процедура Обмен(Команда)
        ОбменНаСервере();
КонецПроцедуры

&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
        Адрес = "http://172.20.10.7/base/ws/ws1.1cws?wsdl";
КонецПроцедуры


// Обмен данными с мобильным приложением
Функция ВыполнитьОбмен(КодУстройства, ДанныеУстройства)
        УстановитьПривилегированныйРежим(Истина);

        УзелОбмена = ПланыОбмена.Мобильные.ЭтотУзел().ПолучитьОбъект();
        Если Не ЗначениеЗаполнено(УзелОбмена.Код) Тогда
                УзелОбмена.Код = "001";
                УзелОбмена.Наименование = "Центральный";
                УзелОбмена.Записать();
        КонецЕсли;

        УзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду(КодУстройства);
        Если УзелОбмена.Пустая() Тогда
                НовыйУзел = ПланыОбмена.Мобильные.СоздатьУзел();
                НовыйУзел.Код = КодУстройства;
                НовыйУзел.Наименование = КодУстройства;
                НовыйУзел.Записать();
                ЗарегистрироватьИзмененияДанных(НовыйУзел.Ссылка);
                УзелОбмена = НовыйУзел.Ссылка;
        КонецЕсли;

        УзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду(КодУстройства);

        ЧтениеXML = Новый ЧтениеXML;
        ЧтениеXML.УстановитьСтроку(ДанныеУстройства.Получить());

        ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
        ЧтениеСообщения.НачатьЧтение(ЧтениеXML);
        ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
ЧтениеСообщения.НомерПринятого);

        НачатьТранзакцию();

        Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
                Данные = ПрочитатьXML(ЧтениеXML);
                Если Не Данные = Неопределено Тогда
                        Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
```

```
                    Данные.ОбменДанными.Загрузка = Истина;
                    Данные.Записать();
                КонецЕсли;
        КонецЦикла;

        ЗафиксироватьТранзакцию();

        ЧтениеСообщения.ЗакончитьЧтение();
        ЧтениеXML.Закрыть();
        УзелОбмена = ПланыОбмена.Мобильные.НайтиПоКоду(КодУстройства);

        ЗаписьXML = Новый ЗаписьXML;
        ЗаписьXML.УстановитьСтроку("UTF-8");
        ЗаписьXML.ЗаписатьОбъявлениеXML();

        ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
        ЗаписьСообщения.НачатьЗапись(ЗаписьXML, УзелОбмена);

        ЗаписьXML.ЗаписатьСоответствиеПространстваИмен("xsi", "http://www.w3.org/2001/XMLSchema-
instanse");
        ЗаписьXML.ЗаписатьСоответствиеПространстваИмен("v8", "http://v8.1c.ru/data");

        ВыборкаИзменений = ПланыОбмена.ВыбратьИзменения(УзелОбмена,
ЗаписьСообщения.НомерСообщения);

        Пока ВыборкаИзменений.Следующий() Цикл
                Данные = ВыборкаИзменений.Получить();
                ЗаписатьXML(ЗаписьXML, Данные);
        КонецЦикла;

        ЗаписьСообщения.ЗакончитьЗапись();
        Возврат Новый ХранилищеЗначения(ЗаписьXML.Закрыть(), новый СжатиеДанных());

КонецФункции

Функция ЗарегистрироватьИзмененияДанных(УзелОбмена) Экспорт
        СоставПланаОбмена = УзелОбмена.Метаданные().Состав;
        Для Каждого ЭлементаСоставаПланаОбмена Из СоставПланаОбмена Цикл
                ПланыОбмена.ЗарегистрироватьИзменения(УзелОбмена,
ЭлементаСоставаПланаОбмена.Метаданные);
        КонецЦикла;
КонецФункции

// Автоматизированное тестирование на ПК
&НаКлиенте
Процедура Тест(Команда)
        АдресEXE = "C:\Program Files (x86)\1cv8\8.3.13.1513\bin\1cv8.exe";
        АдресБазы = "C:\Diplom";
        ЗапуститьСистему(АдресEXE + " ENTERPRISE /F " + АдресБазы + " /TESTCLIENT");
        ТестовоеПриложение = Новый ТестируемоеПриложение("localhost");
        ВремяОкончания = ТекущаяДата() + 60;
        Подключен = Ложь;
        Ошибка = "";
        Пока Не ТекущаяДата() >= ВремяОкончания Цикл
                Попытка
                        ТестовоеПриложение.УстановитьСоединение();
                        Подключен = Истина;
```

```
                Прервать;
            Исключение Ошибка = ОписаниеОшибки();
            КонецПопытки;
        КонецЦикла;

        Если не Подключен Тогда ТестовоеПриложение = Неопределено;
            Сообщить("Не установлено" + Ошибка);
            Возврат;
        КонецЕсли;

        ПроцедураТеста(ТестовоеПриложение);
КонецПроцедуры

&НаКлиенте
Процедура ПроцедураТеста(ТестовоеПриложение)
        ОкноПриложения =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеОкноКлиентскогоПриложения"), "Diplom", , 30);
        Если ВариантТеста = "TC_01" Тогда
            ДатаНачала = ТекущаяУниверсальнаяДатаВМиллисекундах();
            Для Каждого НомерНачала = 1 По 100 Цикл
                // Получение формы тестируемого справочника "Исполнители"

        ОкноПриложения.ВыполнитьКоманду("e1cib/list/Справочник.Исполнители.ФормаСписка");

                // Нажимаем кнопку "Создать"
                КнопкаСоздать =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Создать");
                КнопкаСоздать.Активизировать();
                КнопкаСоздать.Нажать();

                // Заполнение поля "Имя"
                ПолеИмя = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"),
"Имя");
                ПолеИмя.Активизировать();
                ПолеИмя.ВвестиТекст(СформированноеИмя);

                // Заполнение поля "Фамилия"
                ПолеФамилия =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Фамилия");
                ПолеФамилия.Активизировать();
                ПолеФамилия.ВвестиТекст(СформированнаяФамилия);

                // Заполнение поля "Отчество"
                ПолеОтчество =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Отчество");
                ПолеОтчество.Активизировать();
                ПолеОтчество.ВвестиТекст(СформированноеОтчество);

                // Нажимаем кнопку "Записать и закрыть"
                КнопкаЗаписатьИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Записать и закрыть");
                КнопкаЗаписатьИЗакрыть.Активизировать();
                КнопкаЗаписатьИЗакрыть.Нажать();
            КонецЦикла;

            Для Каждого НомерНачала = 1 По 100 Цикл
                // Получение формы тестируемого справочника "Задачи"
```

```
ОкноПриложения.ВыполнитьКоманду("e1cib/list/Справочник.Задачи.ФормаСписка");

                    // Нажимаем кнопку "Создать"
                    КнопкаСоздать =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Создать");
                    КнопкаСоздать.Активизировать();
                    КнопкаСоздать.Нажать();

                    // Заполнение поля "Наименование"
                    ПолеНаименование =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Наименование");
                    ПолеФамилия.Активизировать();
                    ПолеФамилия.ВвестиТекст(СформированнаяФамилия);

                    // Заполнение поля "Крайняя дата внедрения"
                    ПолеДатаВнедрения =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Крайняя дата внедрения");
                    ПолеДатаВнедрения.Активизировать();
                    ПолеДатаВнедрения.ВвестиТекст(СформированнаяДата);

                    // Нажимаем кнопку "Записать и закрыть"
                    КнопкаЗаписатьИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Записать и закрыть");
                    КнопкаЗаписатьИЗакрыть.Активизировать();
                    КнопкаЗаписатьИЗакрыть.Нажать();
                КонецЦикла;

                Для Каждого НомерНачала = 1 По 5 Цикл
                    // Получение формы тестируемого документа "Сертификаты"

ОкноПриложения.ВыполнитьКоманду("e1cib/list/Документ.Сертификаты.ФормаДокумента");

                    //Создание и проведение документов "Сертификаты"
                    ПолеСотрудник =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Наименование");
                    ПолеСотрудник.Активизировать();
                    ПолеСотрудник.Выбрать();
                    ПолеСотрудник.ВыпадающийСписокОткрыт()

ПолеСотрудник.ВыполнитьВыборИзВыпадающегоСписка(СформированныйСотрудник);

                    ПолеИмяДокумента =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Имя документа");
                    ПолеИмяДокумента.Активизировать();
                    ПолеИмяДокумента.ВвестиТекст(СформированноеИмяДокумента);

                    // Нажимаем кнопку "Провести и закрыть"
                    КнопкаПровестиИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Провести и закрыть");
                    КнопкаПровестиИЗакрыть.Активизировать();
                    КнопкаПровестиИЗакрыть.Нажать();
                КонецЦикла;

                Для Каждого НомерНачала = 1 По 100 Цикл
                    // Получение формы тестируемого документа "Сертификаты"
```

```
        ОкноПриложения.ВыполнитьКоманду("e1cib/list/Документ.ИзменениеСтатусаЗадачи.ФормаДокумента");

                        //Создание и проведение документов "Изменение статуса задачи"
                        ПолеЗадача = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Задача");
                        ПолеЗадача.Активизировать();
                        ПолеЗадача.Выбрать();
                        ПолеЗадача.ВыпадающийСписокОткрыт()
                        ПолеЗадача.ВыполнитьВыборИзВыпадающегоСписка(СформированнаяЗадача);

                        ПолеСтатус = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Статус");
                        ПолеСтатус.Активизировать();
                        ПолеЗадача.Выбрать();
                        ПолеЗадача.ВыпадающийСписокОткрыт()
                        ПолеЗадача.ВыполнитьВыборИзВыпадающегоСписка(СформированныйСтатус);

                        // Нажимаем кнопку "Провести и закрыть"
                        КнопкаПровестиИЗакрыть = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Провести и закрыть");
                        КнопкаПровестиИЗакрыть.Активизировать();
                        КнопкаПровестиИЗакрыть.Нажать();
                КонецЦикла;
                ДатаОкончания = ТекущаяУниверсальнаяДатаВМиллисекундах();

        ИначеЕсли ВариантТеста = "TC_02" Тогда
                ДатаНачала = ТекущаяУниверсальнаяДатаВМиллисекундах();

                Для Каждого НомерНачала = 1 По 500 Цикл
                        // Получение формы тестируемого справочника "Исполнители"

        ОкноПриложения.ВыполнитьКоманду("e1cib/list/Справочник.Исполнители.ФормаСписка");

                        // Нажимаем кнопку "Создать"
                        КнопкаСоздать = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Создать");
                        КнопкаСоздать.Активизировать();
                        КнопкаСоздать.Нажать();

                        // Заполнение поля "Имя"
                        ПолеИмя = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Имя");
                        ПолеИмя.Активизировать();
                        ПолеИмя.ВвестиТекст(СформированноеИмя);

                        // Заполнение поля "Фамилия"
                        ПолеФамилия = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Фамилия");
                        ПолеФамилия.Активизировать();
                        ПолеФамилия.ВвестиТекст(СформированнаяФамилия);

                        // Заполнение поля "Отчество"
                        ПолеОтчество = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Отчество");
                        ПолеОтчество.Активизировать();
```

```
                ПолеОтчество.ВвестиТекст(СформированноеОтчество);

                // Нажимаем кнопку "Записать и закрыть"
                КнопкаЗаписатьИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Записать и закрыть");
                КнопкаЗаписатьИЗакрыть.Активизировать();
                КнопкаЗаписатьИЗакрыть.Нажать();
        КонецЦикла;

        Для Каждого НомерНачала = 1 По 1000 Цикл
                // Получение формы тестируемого справочника "Задачи"

    ОкноПриложения.ВыполнитьКоманду("e1cib/list/Справочник.Задачи.ФормаСписка");

                // Нажимаем кнопку "Создать"
                КнопкаСоздать =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Создать");
                КнопкаСоздать.Активизировать();
                КнопкаСоздать.Нажать();

                // Заполнение поля "Наименование"
                ПолеНаименование =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Наименование");
                ПолеФамилия.Активизировать();
                ПолеФамилия.ВвестиТекст(СформированнаяФамилия);

                // Заполнение поля "Крайняя дата внедрения"
                ПолеДатаВнедрения =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Крайняя дата внедрения");
                ПолеДатаВнедрения.Активизировать();
                ПолеДатаВнедрения.ВвестиТекст(СформированнаяДата);

                // Нажимаем кнопку "Записать и закрыть"
                КнопкаЗаписатьИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Записать и закрыть");
                КнопкаЗаписатьИЗакрыть.Активизировать();
                КнопкаЗаписатьИЗакрыть.Нажать();
        КонецЦикла;

        Для Каждого НомерНачала = 1 По 10 Цикл
                // Получение формы тестируемого документа "Сертификаты"

    ОкноПриложения.ВыполнитьКоманду("e1cib/list/Документ.Сертификаты.ФормаДокумента");

                //Создание и проведение документов "Сертификаты"
                ПолеСотрудник =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Наименование");
                ПолеСотрудник.Активизировать();
                ПолеСотрудник.Выбрать();
                ПолеСотрудник.ВыпадающийСписокОткрыт()

    ПолеСотрудник.ВыполнитьВыборИзВыпадающегоСписка(СформированныйСотрудник);

                ПолеИмяДокумента =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Имя документа");
                ПолеИмяДокумента.Активизировать();
                ПолеИмяДокумента.ВвестиТекст(СформированноеИмяДокумента);
```

```
                        // Нажимаем кнопку "Провести и закрыть"
                        КнопкаПровестиИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Провести и закрыть");
                        КнопкаПровестиИЗакрыть.Активизировать();
                        КнопкаПровестиИЗакрыть.Нажать();
                КонецЦикла;

                Для Каждого НомерНачала = 1 По 100 Цикл
                        // Получение формы тестируемого документа "Изменение статуса задачи"

        ОкноПриложения.ВыполнитьКоманду("e1cib/list/Документ.ИзменениеСтатусаЗадачи.ФормаДокуме
нта");
                        //Создание и проведение документов "Изменение статуса задачи"
                        ПолеЗадача = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"),
"Задача");
                        ПолеЗадача.Активизировать();
                        ПолеЗадача.Выбрать();
                        ПолеЗадача.ВыпадающийСписокОткрыт()
                        ПолеЗадача.ВыполнитьВыборИзВыпадающегоСписка(СформированнаяЗадача);

                        ПолеСтатус = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"),
"Статус");
                        ПолеСтатус.Активизировать();
                        ПолеЗадача.Выбрать();
                        ПолеЗадача.ВыпадающийСписокОткрыт()
                        ПолеЗадача.ВыполнитьВыборИзВыпадающегоСписка(СформированныйСтатус);

                        // Нажимаем кнопку "Провести и закрыть"
                        КнопкаПровестиИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Провести и закрыть");
                        КнопкаПровестиИЗакрыть.Активизировать();
                        КнопкаПровестиИЗакрыть.Нажать();
                КонецЦикла;
                ДатаОкончания = ТекущаяУниверсальнаяДатаВМиллисекундах();

        ИначеЕсли ВариантТеста = "TC_03" Тогда
                ДатаНачала = ТекущаяУниверсальнаяДатаВМиллисекундах();

                Для Каждого НомерНачала = 1 По 1000 Цикл
                        // Получение формы тестируемого справочника "Исполнители"

        ОкноПриложения.ВыполнитьКоманду("e1cib/list/Справочник.Исполнители.ФормаСписка");

                        // Нажимаем кнопку "Создать"
                        КнопкаСоздать =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Создать");
                        КнопкаСоздать.Активизировать();
                        КнопкаСоздать.Нажать();

                        // Заполнение поля "Имя"
                        ПолеИмя = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"),
"Имя");
                        ПолеИмя.Активизировать();
                        ПолеИмя.ВвестиТекст(СформированноеИмя);

                        // Заполнение поля "Фамилия"
```

```
                ПолеФамилия =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Фамилия");
                ПолеФамилия.Активизировать();
                ПолеФамилия.ВвестиТекст(СформированнаяФамилия);

                // Заполнение поля "Отчество"
                ПолеОтчество =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Отчество");
                ПолеОтчество.Активизировать();
                ПолеОтчество.ВвестиТекст(СформированноеОтчество);

                // Нажимаем кнопку "Записать и закрыть"
                КнопкаЗаписатьИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Записать и закрыть");
                КнопкаЗаписатьИЗакрыть.Активизировать();
                КнопкаЗаписатьИЗакрыть.Нажать();

        КонецЦикла;
        Для Каждого НомерНачала = 1 По 10000 Цикл
                // Получение формы тестируемого справочника "Задачи"

    ОкноПриложения.ВыполнитьКоманду("e1cib/list/Справочник.Задачи.ФормаСписка");

                // Нажимаем кнопку "Создать"
                КнопкаСоздать =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Создать");
                КнопкаСоздать.Активизировать();
                КнопкаСоздать.Нажать();

                // Заполнение поля "Наименование"
                ПолеНаименование =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Наименование");
                ПолеФамилия.Активизировать();
                ПолеФамилия.ВвестиТекст(СформированнаяФамилия);

                // Заполнение поля "Крайняя дата внедрения"
                ПолеДатаВнедрения =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Крайняя дата внедрения");
                ПолеДатаВнедрения.Активизировать();
                ПолеДатаВнедрения.ВвестиТекст(СформированнаяДата);

                // Нажимаем кнопку "Записать и закрыть"
                КнопкаЗаписатьИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Записать и закрыть");
                КнопкаЗаписатьИЗакрыть.Активизировать();
                КнопкаЗаписатьИЗакрыть.Нажать();
        КонецЦикла;

        Для Каждого НомерНачала = 1 По 15 Цикл
                // Получение формы тестируемого документа "Сертификаты"

    ОкноПриложения.ВыполнитьКоманду("e1cib/list/Документ.Сертификаты.ФормаДокумента");

                //Создание и проведение документов "Сертификаты"
                ПолеСотрудник =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Наименование");
                ПолеСотрудник.Активизировать();
```

```
                ПолеСотрудник.Выбрать();
                ПолеСотрудник.ВыпадающийСписокОткрыт()
            ПолеСотрудник.ВыполнитьВыборИзВыпадающегоСписка(СформированныйСотрудник);
                ПолеИмяДокумента =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"), "Имя документа");
                ПолеИмяДокумента.Активизировать();
                ПолеИмяДокумента.ВвестиТекст(СформированноеИмяДокумента);
                // Нажимаем кнопку "Провести и закрыть"
                КнопкаПровестиИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Провести и закрыть");
                КнопкаПровестиИЗакрыть.Активизировать();
                КнопкаПровестиИЗакрыть.Нажать();
            КонецЦикла;

            Для Каждого НомерНачала = 1 По 200 Цикл
                // Получение формы тестируемого документа "изменение статуса задачи"
        ОкноПриложения.ВыполнитьКоманду("e1cib/list/Документ.ИзменениеСтатусаЗадачи.ФормаДокуме
нта");
                //Создание и проведение документов "Изменение статуса задачи"
                ПолеЗадача = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"),
"Задача");
                ПолеЗадача.Активизировать();
                ПолеЗадача.Выбрать();
                ПолеЗадача.ВыпадающийСписокОткрыт()
                ПолеЗадача.ВыполнитьВыборИзВыпадающегоСписка(СформированнаяЗадача);
                ПолеСтатус = ТестовоеПриложение.НайтиОбъект(Тип("ТестируемоеПолеФормы"),
"Статус");
                ПолеСтатус.Активизировать();
                ПолеЗадача.Выбрать();
                ПолеЗадача.ВыпадающийСписокОткрыт()
                ПолеЗадача.ВыполнитьВыборИзВыпадающегоСписка(СформированныйСтатус);
                // Нажимаем кнопку "Провести и закрыть"
                КнопкаПровестиИЗакрыть =
ТестовоеПриложение.НайтиОбъект(Тип("ТестируемаяКнопкаФормы"), "Провести и закрыть");
                КнопкаПровестиИЗакрыть.Активизировать();
                КнопкаПровестиИЗакрыть.Нажать();
            КонецЦикла;
            ДатаОкончания = ТекущаяУниверсальнаяДатаВМиллисекундах();
        КонецЕсли;
        ВремяВыполнения = ДатаОкончания - ДатаНачала;
КонецПроцедуры

// Тестирование мобильного клиента
&НаСервере
Процедура СоздатьИсполнителей(Количество)
        Для Каждого Исп = 1 По Количество Цикл
            НовыйИсполнитель = Справочники.Исполнители.СоздатьЭлемент();
            НовыйИсполнитель.Фамилия = СформированнаяФамимлия;
            НовыйИсполнитель.Имя = СформированноеИмя;
            НовыйИсполнитель.Отчество = СформированноеОтчество;
            НовыйИсполнитель.ОбластьСпециализации = СформированнаяОбластьСпециализации;
            НовыйИсполнитель.Записать();
        КонецЦикла;
КонецПроцедуры

&НаСервере
Процедура СоздатьЗадачи(Количество)
```

```
	Для Каждого Задача = 1 По Количество Цикл
		НоваяЗадача = Справочники.Исполнители.СоздатьЭлемент();
		НоваяЗадача.Наименование = СформированноеНаименование;
		НоваяЗадача.КрайняяДатаВнедрения = СформированнаяДата;
		НоваяЗадача.Записать();
	КонецЦикла;
КонецПроцедуры

&НаСервере
Процедура ПровестиСертификаты(Количество)
	Для Каждого Сертификат = 1 По Количество Цикл
		НовыйСертификат = Документы.Сертификаты.СоздатьДокумент();
		НовыйСертификат.Сотрудник =
Справочники.Исполнители.НайтиПоНаименованию(СформированныйИсполнитель);
		НовыйСертификат.СрокДействия = СформированнаяДатаДействия;
		НовыйСертификат.Записать(РежимЗаписиДокумента.Проведение);
	КонецЦикла;
КонецПроцедуры

&НаСервере
Процедура ПровестиЗадачи(Количество)
	 Для Каждого Сертификат = 1 По Количество Цикл
		НовоеИзменениеСтатуса = Документы.Сертификаты.СоздатьДокумент();
		НовоеИзменениеСтатуса.Задача =
Справочники.Задачи.НайтиПоНаименованию(СформированнаяЗадача);
		НовоеИзменениеСтатуса.Статус = СформированныйСтатус;
		НовоеИзменениеСтатуса.Записать(РежимЗаписиДокумента.Проведение);
	КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура
	Если ВариантТеста = "ТС_01" Тогда
		ДатаНачала = ТекущаяУниверсальнаяДатаВМиллисекундах();
		СоздатьИсполнителей(100);
		СоздатьЗадачи(100);
		ПровестиСертификаты(5);
		ПровестиЗадачи(100);
		ДатаОкончания = ТекущаяУниверсальнаяДатаВМиллисекундах();
	ИначеЕсли ВариантТеста = "ТС_02" Тогда
		ДатаНачала = ТекущаяУниверсальнаяДатаВМиллисекундах();
		СоздатьИсполнителей(500);
		СоздатьЗадачи(1000);
		ПровестиСертификаты(10);
		ПровестиЗадачи(100);
		ДатаОкончания = ТекущаяУниверсальнаяДатаВМиллисекундах();
	ИначеЕсли ВариантТеста = "ТС_03" Тогда
		ДатаНачала = ТекущаяУниверсальнаяДатаВМиллисекундах();
		СоздатьИсполнителей(1000);
		СоздатьЗадачи(10000);
		ПровестиСертификаты(200);
		ПровестиЗадачи(15);
		ДатаОкончания = ТекущаяУниверсальнаяДатаВМиллисекундах();
	КонецЕсли;
	Результат = ДатаОкончания - ДатаНачала;
КонецПроцедуры
```