# Содержание

[Введение 5](#_Toc70360012)

[1 Обзор предметной области 7](#_Toc70360013)

[2 Постановка задачи 13](#_Toc70360014)

[3 Проектирование приложения 14](#_Toc70360015)

[3.1 Проектирование мобильной части приложения 14](#_Toc70360016)

[3.2 Проектирование WEB части приложения 25](#_Toc70360016)

[3.3 Проектирование базы данных приложения 31](#_Toc70360017)

[3.4 Описание таблиц базы данных 33](#_Toc70360018)

[Заключение 3](#_Toc70360019)6

[Перечень использованных информационных ресурсов 37](#_Toc70360020)

[Приложение А](#_Toc70360021) [Техническое задание 38](#_Toc70360022)

[Приложение Б](#_Toc70360023) [Листинг программы 47](#_Toc70360024)

# Введение

Для любого предприятия, в независимости от его размера, одним из ключевых факторов успешной работы и дальнейшего роста являться своевременное и качественное управление персоналом. Данное утверждение хорошо доказывается на примере успешных фирм (Apple, Toyota, Huawue и т.д.), где во главу угла ставят менеджмент персонала.

На данный момент существует множества методологий того, как эффективно управлять персоналом, самые популярные и пользующиеся спросам это Scrum и Kanban. Это гибкие методологии создания продукта. По ним можно работать в любой отрасли, но особенно хорошо они подходят для ИТ. В основе обеих методологий лежат принципы Agile.

Сам Agile (agile software development, от англ. agile – проворный) – это семейство «гибких» подходов к разработке программного обеспечения. Такие подходы также иногда называют фреймворками или agile-методологиями [1].

Краткое смысловое содержание Agile сформулировано в Agile-манифесте: «Люди и взаимодействие важнее процессов и инструментов. Работающий продукт важнее исчерпывающей документации. Сотрудничество с заказчиком важнее согласования условий контракта. Готовность к изменениям важнее следования первоначальному плану» [1].

Для того чтобы можно было воплотить в жизнь данные методологии нужно соответствующее программное решение, в котором важную роль будит играть пользовательский интерфейс.

Таким образом, целью данной работы является автоматизация процессов распределения и контроля задача сотрудниками предприятия, внедрение которого позволит:

* планирования задач для сотрудников на день;
* следить за успешностью выполнения поставленных задач;
* вести статистику успеваемости каждого сотрудника отдельно;
* просмотр общей статистики по всем задачам на день;
* перераспределение не выполненных задач на новый рабочий день;
* возможность гибкого планирования.

1 Обзор предметной области

Управление персоналом - сфера деятельности руководящего состава организации, направленная на повышение эффективности работы организации за счет повышения эффективности работы с ее сотрудниками психологическими, правовыми, экономическими и другими методами [2].

Управление персоналом заключатся:

* в формировании системы управления персоналом;
* в планировании кадровой работы;
* в проведении маркетинга персонала;
* в определении кадрового потенциала и потребности организации в персонале.

Возрастает роль и значение кадровых служб, происходит их интеграция в единые структуры управления персоналом, расширяются выполняемые ими функции.

В настоящее время среди отечественных специалистов нет единого мнения относительно сущности и содержания термина "управление персоналом". Зачастую его смешивают с терминами "кадровая политика" либо "кадровая работа". Необходимо, однако, отметить, что управление персоналом как отрасль менеджмента имеет иной объект, иные структуру и содержание, поэтому более корректно, говорить об управлении персоналом как о комплексной социальной технологии (системе технологий), направленной на повышение эффективности использования человеческих ресурсов предприятия.

Существующие программные продукты для контроля персонала рассчитаны на мониторинг работы офисных сотрудников. Их можно разделить на две категории: традиционное программное обеспечение и онлайн-сервисы, то есть программное обеспечение как услуга.

Рассмотрим принцип работы современных программных продуктов для контроля персонала, их преимущества и недостатки.

Чтобы начать использовать традиционное программное обеспечение, необходимо приобрести программный пакет и установить его на компьютеры внутри компании. Программа ведет мониторинг работы персонала, фиксирует сайты и файлы, с которыми работают сотрудники и сохраняет полученные данные на внутренних серверах компании.

Преимущества:

* Собранные программой данные хранятся внутри компании. Это подходит для компаний с повышенным уровнем безопасности.
* Работа программы не зависит от интернета — в пределах компании вы всегда можете получить доступ к данным программы.

Недостатки:

* Иногда внедрение программного обеспечения занимает много времени.
* Часто, собранные программой данные, предоставляются в виде простого списка. Это очень неудобно для анализа.
* Некоторые программы негативно влияют на производительность компьютера и тормозят его работу.
* Локальность программы не позволяет работать с ней удаленно.
* Высокая стоимость (программный пакет, доплата за обновления, дополнительные траты на специалистов для обслуживания).

Онлайн-сервисы – предоставляют возможность использовать приложение через веб-браузер. Говоря простым языком, вы не приобретаете программу, а берёте её в аренду. Вы не тратите деньги и время на её обслуживание, не тратите деньги на обновления потому, что они бесплатны. Просто оплачиваете аренду и пользуетесь сервисом и всеми преимуществами, которые он дает.

Преимущества онлайн-сервиса:

* Устанавливается легко и быстро.
* Собранные сервисом данные предоставляются в виде удобных для анализа отчетов.
* Не влияет на производительность компьютера.
* Доступен в любое время, из любой точки планеты.
* Помесячная оплата делает сервис доступным для небольших и средних компаний.

Недостатки онлайн-сервиса:

* Не подходит для компаний, система безопасности которых требует локального хранения всех данных.
* Зависимость от интернета – если отсутствует сеть, вы не можете получить доступ к отчетам о работе сотрудников.

После данного краткого обзора становиться очевидным решение производить дальнейшую разработку на основе онлайн-сервеса.

Проведем анализ существующих систем, отвечающих за контроль над персоналом: «WEEEK», «Things», «Habitica».

В основе WEEEK принцип недельного планирования — расписываешь планы на неделю и последовательно двигаешься от задачи к задаче. Можно создавать рабочие области (воркспейсы), в них — проекты. У задач можно делать подзадачи до 6 уровней вложенности — это будут полноценные задачи, а не простые чек-листы (хотя, и так их можно тоже использовать). В WEEEK есть мощная и гибкая система уведомлений — ты можешь самостоятельно указать, где какие уведомления ты хочешь получать.

Может работать на таких платформах как: веб, Windows, macOS, iOS, Android.

Плюсы:

* понятный интерфейс;
* нет ограничений по возможностям;
* удобные мобильные приложения;
* удобная система уведомлений.

Минусы:

* в веб-версии пока нельзя смотреть задачи в виде списка (только доски или календарь);
* в мобильных приложениях пока не реализованы все возможности.

На рисунке 1 представлен вид главной страницы приложения «WEEEK».

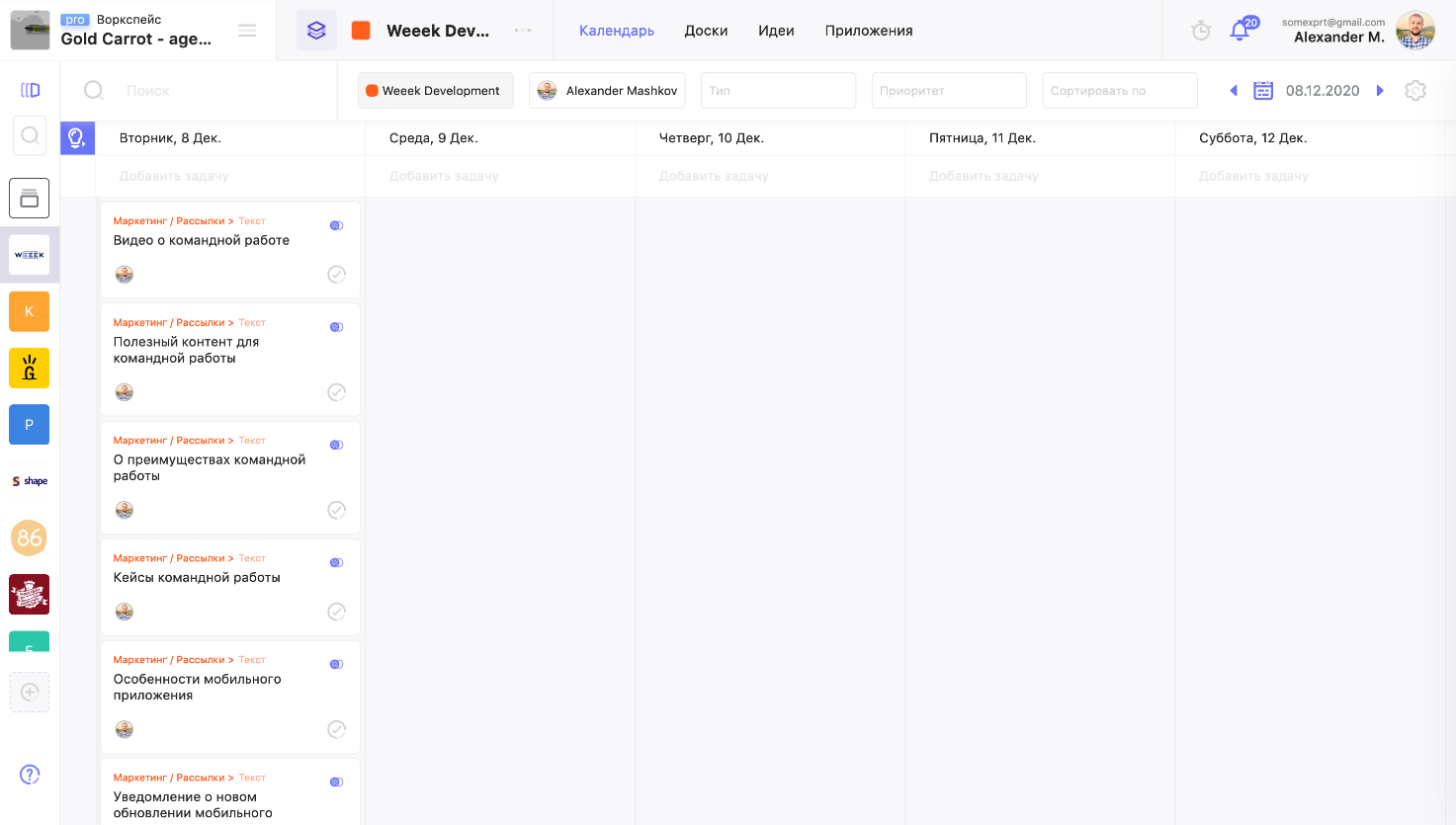


Рисунок 1 – Главная страница приложения «WEEEK»

Things — это проект немецкой команды в виде приложений для iPhone, iPad и Mac. Дизайн вполне приемлем для приложения такого рода, панель с папками расположена в левой части, дела делятся по категориям на «Сегодня», «Следующие», «Запланировано», «Потом» и «Проекты».

Кроме того, задачи можно вносить из других программ, к примеру, из браузера, выполненные задания переносятся в «Журнал», при удалении — в «Корзину». К сожалению, отсутствуют напоминания.

Может работать на таких платформах как: macOS, iOS.

Плюсы:

* простой и приятный интерфейс;
* отсутствие лишних деталий;
* удобная система с разделом «Сегодня».

Минусы:

* большая стоимость;
* продаётся на каждую платформу отдельно;
* есть только на устройствах Apple.

На рисунке 2 представлен вид главной страницы приложения «Things».

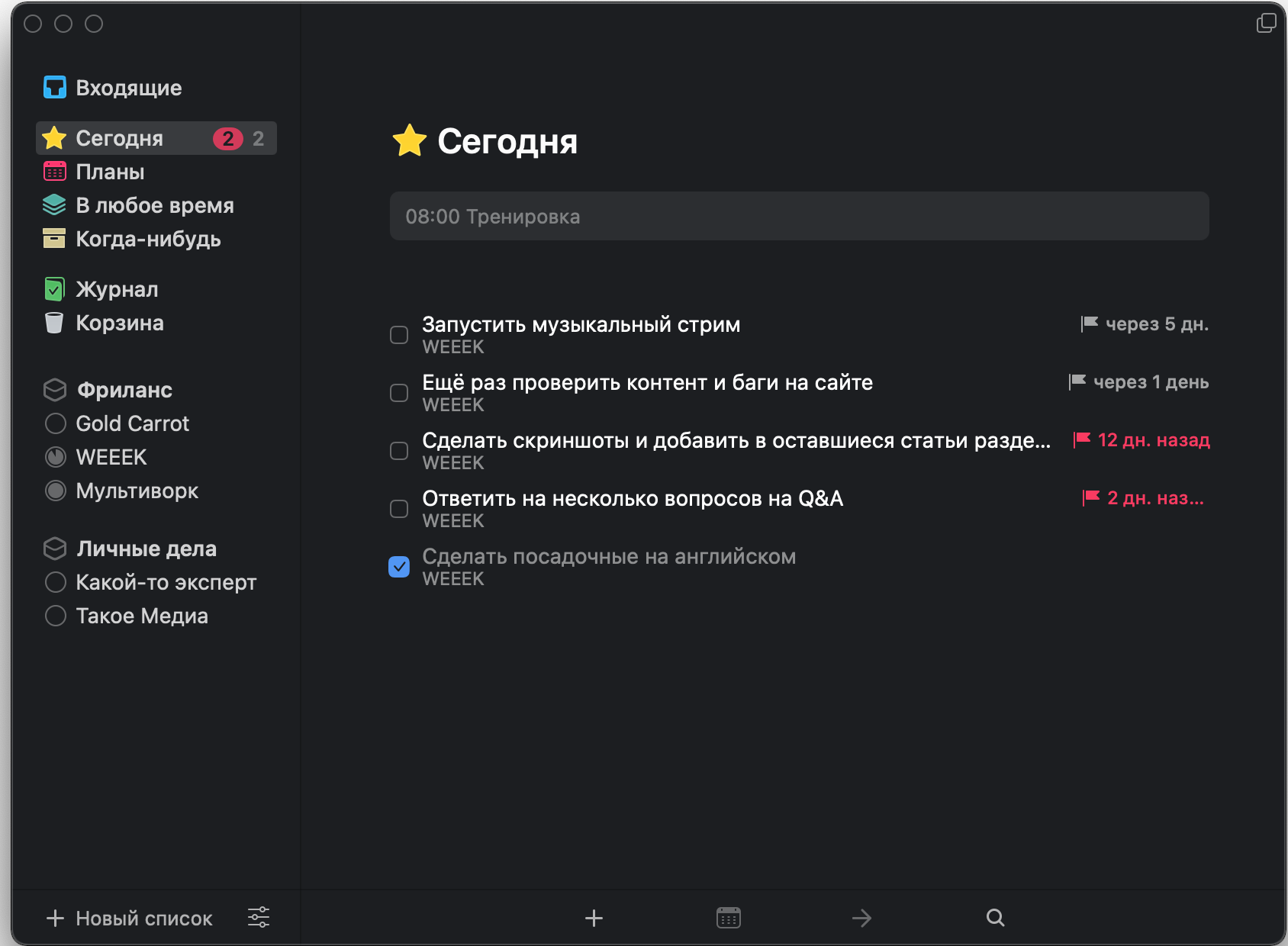


Рисунок 2 – Главная страница приложения «Things»

Habitica, которая раньше называлась HabitRPG, использует принципы геймдизайна, чтобы мотивировать тебя делать задачи вовремя. Работает просто: ты создаёшь персонажа, который получает опыт за выполнение задач (твоих реальных задач) и урон за их невыполнение или откладывание.

Плюсы:

* мотивирует делать задачи вовремя;
* приятный и красивый пользовательский интерфейс.

Минусы:

* игровая часть может отвлекать от работы;
* как задачник приложение работает не очень корректно.

На рисунке 3 представлен вид главной страницы приложения «Habitica».

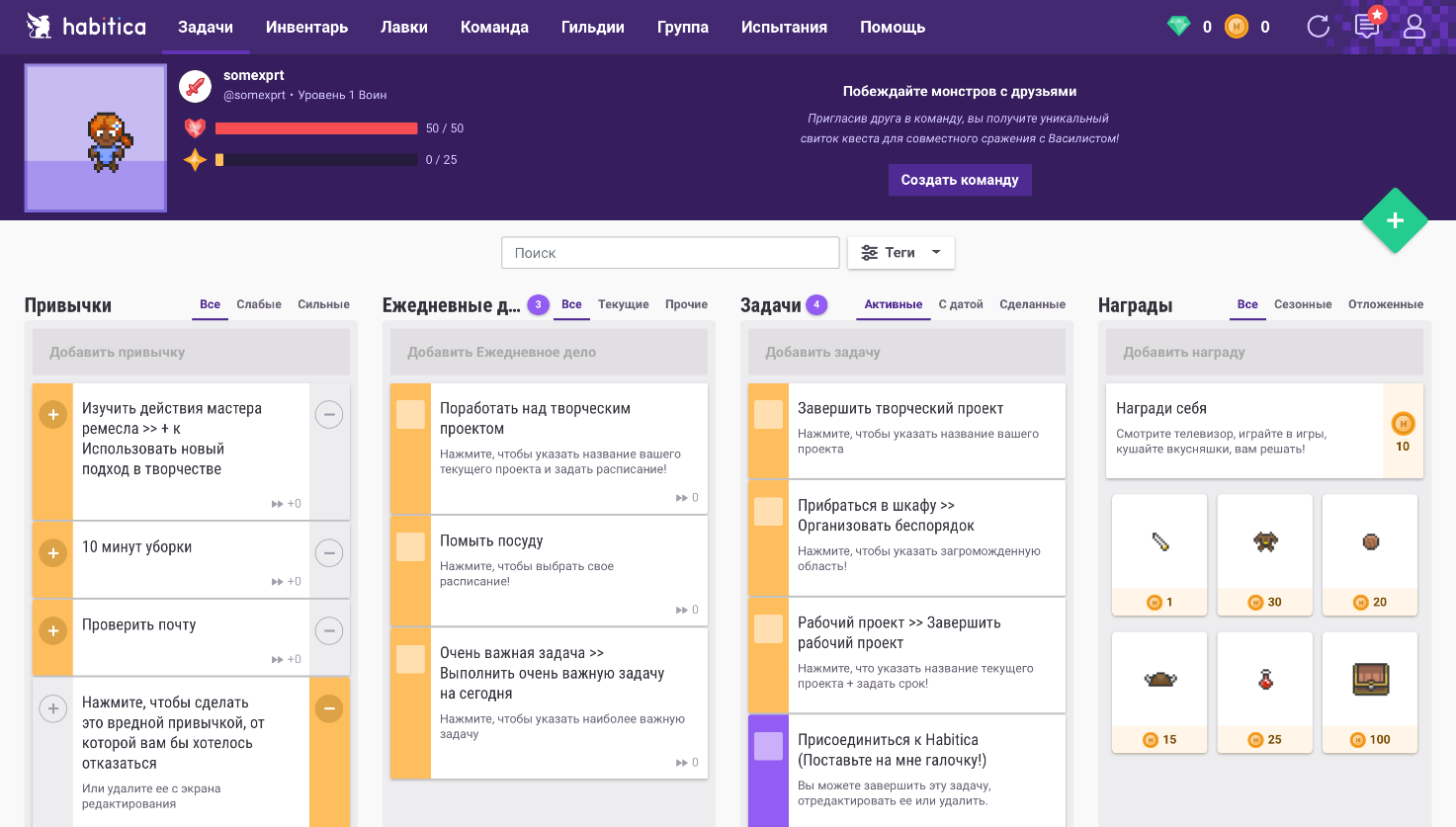


Рисунок 3 – Главная страница приложения «Habitica»

Может работать на таких платформах как: веб, iOS, Android.

2 Постановка задачи

Целью выпускной квалификационной работы является проектирование и разработка системы для дистанционного управления персоналом.

Перечень задач для реализации:

* Анализ предметной области и разработка технического задания.
* Реализация WEB части приложения.
* Реализация мобильной части приложения.
* Проектирование базы данных системы.

Техническое задания можно посмотреть в приложении А.

После просмотра существующих решений данной поставленной задачи был сделан вывод о том, что к основным функциональным характеристикам, которыми должен обладать проект такого рода можно отнести:

* возможность планирования задач менеджером для конкретного сотрудника;
* просмотр менеджером статистики выполнения задач;
* добавления менеджером нового сотрудника в систему;
* удаления менеджером сотрудника из системы;
* изменения данных менеджером о сотруднике в системе;
* просмотр своего списка задач сотрудником;
* возможность сотрудником отметить задачу как выполненную;
* возможность прикрепления отчета о выполненной задачи;
* возможность отмены задачи сотрудником;
* уточнения деталей выполнения задачи.

3 Проектирование приложения

Проектирование системы для контроля за персоналом является многоступенчатым процессом, который создаться с использованием совокупности методологий и инструментальных средств. Выделяя основные этапы можно выделить: инфологическое (концептуальное), даталогическое (логическое) и алгоритмическое проектирование.

3.1 Проектирование мобильной части приложения

В данном разделе мы рассмотрим алгоритмическое и инфологическое (концептуальное) проектирование мобильное части приложения, которое помогает дистанционно управлять персоналом.

Сперва будут рассмотрены основные процессы алгоритмического проектирования с подробным описанием основных алгоритмов данного модуля, которые обеспечивают заявленный функционал, а также будут представлены блок схемы данных алгоритмов.

Всего будут рассмотрены три основных алгоритма это:

* алгоритм выполнения задачи;
* алгоритм уточнения информации о задачи;
* алгоритм прикрепления отчета о выполненной задачи.

Алгоритм выполнения задачи используется в том случае, когда сотрудник хочет сообщить менеджеру о том, что он выполнил поставленную перед ним задачу и состоит из следующих шагов:

Шаг 1. Сотрудник отмечает выполненную им задачу.

Шаг 2. Происходить создания тела запроса, которое будет отправлено на сервер.

Шаг 3. Отправка подготовленного запроса на сервер.

Шаг 4. Если задача было выполнено корректно и по всем правилам и нормам принятым руководящим составом, то происходит переход к шагу 7, иначе происходит переход к шагу 5.

Шаг 5. Отменяется помеченное ранее задание.

Шаг 6. Восстановление состояния, которое было до момента выбора сотрудником задачи, как выполненной и переход к шагу 1.

Шаг 7. Сохранения состояния помеченной задачи в базе данных как выполненной.

Шаг 8. Отображения и обработка текущего состояния.

Блок схема данного алгоритма изображена на рисунке 5.

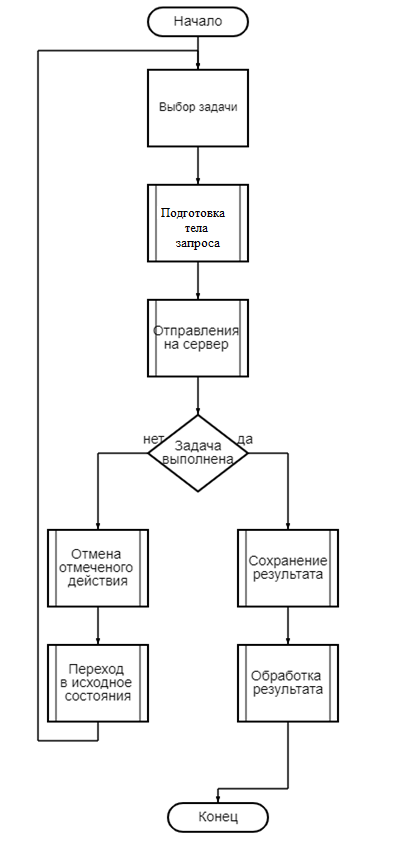


Рисунок 5 – Блок схема алгоритма выполнения задачи

Алгоритм уточнения информации о задаче используется тогда, когда сотруднику нахватает информации о выполнении задачи, которую ему предоставил менеджер и состоит из следующих шагов:

Шаг 1. Сотрудник отмечает задачу, на которую он хочет получить больше информации.

Шаг 2. Сотрудник вводит сообщение, в котором описывает свой вопрос.

Шаг 3. Сотрудник нажимает на кнопку отправки.

Шаг 4. Происходит создания тела запроса, который отправиться на сервер.

Шаг 5. Отменяется помеченное ранее задание.

Шаг 6. Отправка созданного запроса на сервер.

Шаг 7. Ожидание получения ответа от менеджера.

Шаг 8. Если менеджер ответил на запрос сотрудника, то происходит переход на шаг 9, иначе происходит переход на шаг 8.

Шаг 9. Отображения ответа, который дал менеджер.

Блок схема данного алгоритма изображена на рисунке 6.



Рисунок 6 – Блок схема алгоритма уточнения информации о задаче

Алгоритм прикрепления отчета о выполненной задачи используется тогда, когда сотрудник отметил задачу как выполненную и имеет следующие шаги:

Шаг 1. Сотрудник выберете задачу, которую хочет выполнить.

Шаг 2. Сотрудник помечает задачу, как выполненую.

Шаг 3. Если сотрудник захочет прикрепить фото отчет к проделанной работе, то происходит переход к шагу 4, иначе к шагу 6.

Шаг 4. Сотрудник выбирает фото материал, который он хочет представить в виде отчета.

Шаг 5. Выбранный фото материал загружается в базу данных приложения.

Шаг 6. Происходит создания тела запроса, который отправиться на сервер.

Шаг 7. Отправка созданного запроса на сервер.

Шаг 8. Получение статуса задачи от сервера.

Блок схема данного алгоритма изображена на рисунке 7.

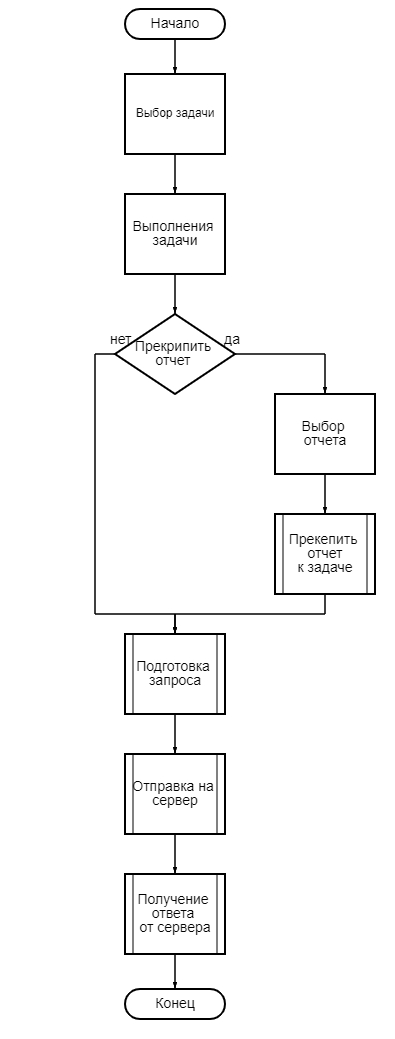


Рисунок 7 – Блок схема алгоритма прикрепления отчета о выполнения задачи

Инфологическая модель представляет собой описание предметной области, которое осуществляется с помощью специальных средств. В данной выпускной квалификационной работа инфологическая модель разработана на основе стандарта языка UML, который используется для спецификации, визуализации, конструирования и документирования системы.

С помощи приставленного выше инструмента проектирования можно описать предметную область в виде UML-диаграммы классов, которая представлена на рисунке 4.

Из UML-диаграммы видна что иерархия мобильной части данного приложения включает пять основных классов (Task, Graph Component, Statistick, Enter, Manager), на которых основывается данный модуль. Так же на диаграмме представлены все имеющиеся виды связи между классами, которые помогают понять способы взаимодействия между ними.

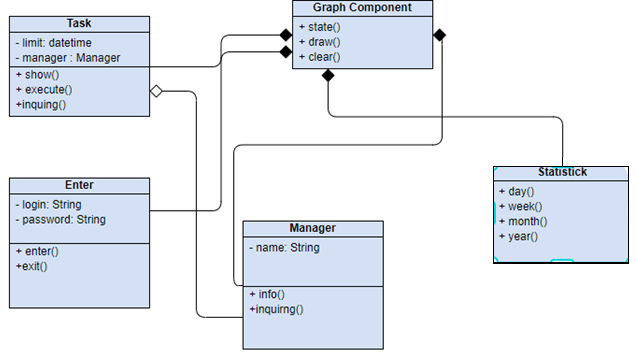


Рисунок 4 – UML диаграмма классов мобильной части приложения

Рассмотрим основные классы данного модуля.

Класс «Task» представляет собой сущность задачи, которую менеджер предварительно поставил сотруднику и имеет следующие методы:

* метод show является публичным, не принимает входных параметров и отвечает за передачу информации о задачи;
* метод execute является публичным, не принимает входных параметров и отвечает за выполнения задачи;
* метод inquing является публичным, не принимает входных параметров и отвечает за способ уточнения о задачи сотрудником.

Так же класс «Task» имеет следующие поля:

* поле limit являться типом данных datetime, оно приватное и отвечает за информацию о сроках выполнения задачи;
* поле manager являться типом данных Manager, оно приватное и отвечает за ссылку на класс менеджера, который поставил данную задачу.

Класс «Enter» представляет собой сущность входа, которая отвечает за корректность легирования (входа) в приложения сотрудником и имеет следующие методы:

* метод enter является публичным, не принимает входных параметров и отвечает за валидацию и корректный вход в мобильные приложения;
* метод exite является публичным, не принимает входных параметров и отвечает за корректный выход из мобильного приложения.

Так же класс «Enter» имеет следующие поля:

* поле login являться типом данных String, оно приватное и отвечает за информацию о логине пользователя;
* поле password являться типом данных String, оно приватное и отвечает за информацию о пароли пользователя.

Класс «Manager»представляет собой сущность менеджера и имеет следующие методы:

* метод info является публичным, не принимает входных параметров и отвечает за передачу информации о менеждере;
* метод inquring является публичным, не принимает входных параметров и отвечает за способ отправления уведомления менеджеру.

Так же класс «Manager» имеет следующие поля:

* поле name являться типом данных String, оно приватное и отвечает за персональную информацию менеджера.

Класс «Staticktick» представляет собой сущность статистики, которую менеджер может просмотреть, для принятия решения кому именно из сотрудников нужно планировать задачу и имеет следующие методы:

* метод day является публичным, не принимает входных параметров и отвечает за информацию о статистики за день;
* метод week является публичным, не принимает входных параметров и отвечает за информацию о статистики за неделю;
* метод month является публичным, не принимает входных параметров и отвечает за информацию о статистики за месяц;
* метод year является публичным, не принимает входных параметров и отвечает за информацию о статистики за год.

Класс «Graph Component» представляет собой сущность, которая отвечает за корректную от рисовки всех компонентов пользовательского интерфейса и имеет следующие методы:

* метод state является публичным, не принимает входных параметров и отвечает за информацию о текущем состоянии отображения;
* метод draw является публичным, не принимает входных параметров и отвечает за корректную отрисовку;
* метод clear является публичным, не принимает входных параметров и отвечает за удаления текущего состояния отображения.

Так как мобильный модуль проектируется на операционной системе Android, то сама разработка и написание кода будит происходить в IDEA Android Studio. Это специальное IDEA для разработки мобильных приложений под Android от компании JetBrains, в ней можно разрабатывать преимущественно на двух языках программирования это Kotlin и Java.

Так как владелец ОС Android компания Google рекомендует писать все новые проекты на Kotlin, связывая это с тем что это более современный язык программирования, который поддерживает не только парадигму объектное ориентированного программирования, но и функционального, а также обладает более выразительным, лаконичным и понятным синтаксисам в отличие от Java, то данный модуль будит написан на языке программирования Kotlin.

Так же необходимо ввести некоторые общие понятия, с которыми предстоит столкнуться на данном этапе проектирования. Важное базовое понятие - это SDK или Software Development Kit («devkit»). Под ним понимают набор средств разработки, позволяющий программистам создавать приложения под некоторую архитектуру - для определенного пакета программ, аппаратной или программной платформы, операционной или компьютерной систем. SDK, как правило, предоставляет разработчику широкий набор инструментов, который может включать в себя средства отладки и справочные материалы (примеры кода, заметки и т.д.). SDK часто предоставляется непосредственно разработчиком целевой платформы и в нем учитываются её особенности. Распространятся (как и в случае с Android) такой инструментарий может бесплатно. Это помогает платформе обзаводиться разнообразными приложениями за счет сторонних разработчиков.

Android SDK требует JDK версии 5 или выше. На сайтах разработчиков доступны версии SDK, для операционных систем Windows, Linux и Mac OS, а в SDK входит эмулятор для каждой из них. Точнее говоря, приложения под Android выполняются в виртуальной машине, и выбор любой из существующих операционных систем не дает никаких преимуществ перед другими, так что можно работать с той, которая более удобна.

3.2 Проектирование WEB части приложения

В данном разделе мы рассмотрим алгоритмическое и инфологическое (концептуальное) проектирование WEB части приложения, которое помогает дистанционно управлять персоналом.

Для начала рассмотрим основные процессы алгоритмического проектирования с подробным описанием основных алгоритмов WEB части приложения, которые обеспечивают заявленный функционал, а также будут представлены блок схемы данных алгоритмов.

Всего будут рассмотрены два основных алгоритма это:

* алгоритм добавления задачи;
* алгоритм просмотра статистике.

Алгоритм добавления задачи используется в том случае, когда менеджер хочет добавить новую задачу и состоит из следующих шагов:

Шаг 1. Менеджер вводит информацию о задачи (заголовок, описание, время выполнения и т. д.).

Шаг 2. Менеджер вводит срок выполнения задачи.

Шаг 3. Назначение менеджером сотрудника для выполнения данной задачи.

Шаг 4. Создание модели задачи.

Шаг 5. Отправка запроса на сервер, тело которого содержит модель задачи.

Шаг 6. Передача модели задачи в шаблон, который корректно от рисует задачу.

Блок схема данного алгоритма изображена на рисунке 9.

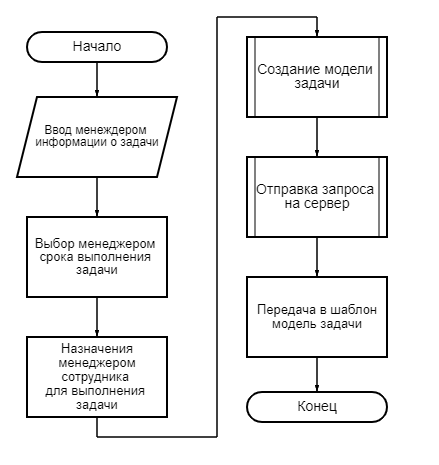


Рисунок 9 – Блок схема алгоритма добавления задачи

Алгоритм просмотра статистики используется в том случае, когда менеджер хочет посмотреть текущею статистику по выполнении задач и состоит из следующих шагов:

Шаг 1. Менеджер выбирает вид статистики (общая, персональная).

Шаг 2. Если менеджер выбрал общую, то переходим к шагу 4, иначе к шагу 3.

Шаг 3. Прикрепляем id выбранного сотрудника и прикрепляем его к телу запроса.

Шаг 4. Отправка запроса на сервер.

Шаг 5. Получение и разбор ответа от сервера.

Шаг 6. Создания модели статистики из полуучёного ответа от сервера.

Шаг 7. Передача в шаблон модели статистики, которая будет корректно от рисована.

Блок схема данного алгоритма изображена на рисунке 10.

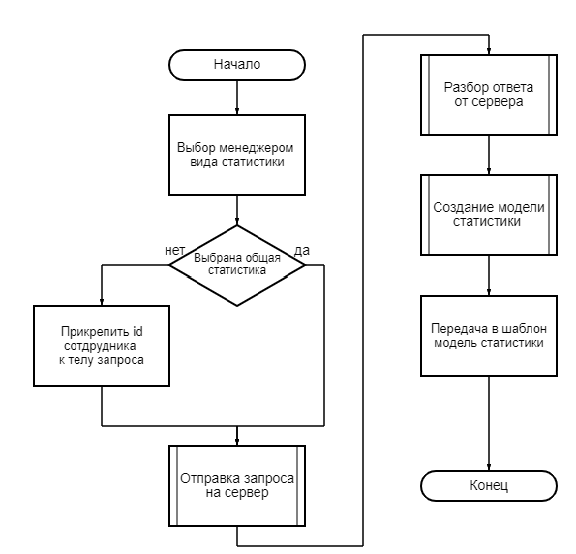


Рисунок 10 – Блок схема алгоритма просмотра статистики

Инфологическая модель данного модуля будит описана при помощи языка UML, который используется для спецификации, визуализации, конструирования и документирования системы.

С помощи приставленного выше инструмента проектирования можно описать предметную область в виде UML-диаграммы классов, которая представлена на рисунке 8.

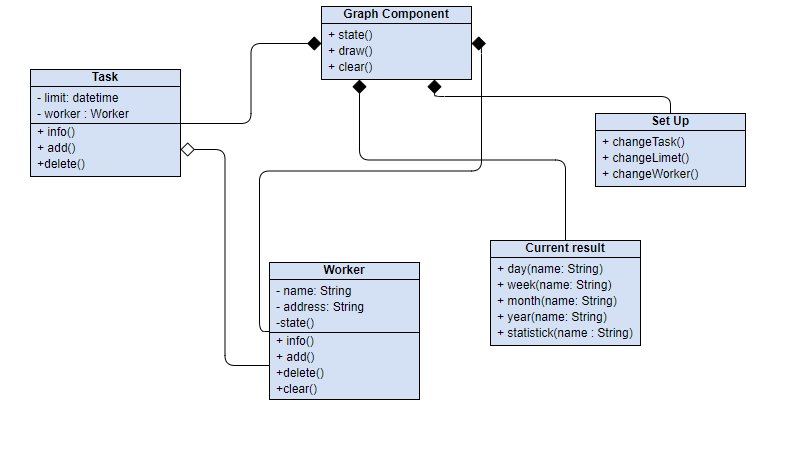


Рисунок 8 – UML диаграмма классов WEB части приложения

Из UML диаграммы видно, что вся предметная область данного модуля заключается в пяти основных классах (Task, Graph Component, Set Up, Worker, Current result). Так же на диаграмме видно все связи между классами, что существуют в данной предметной области.

Рассмотрим базовые классы данного модуля.

Класс «Task» представляет собой сущность задачи, которую менеджер предварительно поставил сотруднику и имеет следующие методы:

* метод info является публичным, не принимает входных параметров и отвечает за представления информации о задачи;
* метод add является публичным, не принимает входных параметров и отвечает за добавления новой задачи;
* метод delete является публичным, не принимает входных параметров и отвечает за удаление существующей задачи.

Так же класс «Task» имеет следующие поля:

* поле limit являться типом данных datetime, оно приватное и отвечает за информацию о сроках выполнения задачи;
* поле worker являться типом данных Worker, оно приватное и отвечает за ссылку на класс сотрудника, к которому прикреплена поставленная задача.

Класс «Graph Component**»** представляет собой сущность, которая отвечает за корректную от рисовки всех компонентов пользовательского интерфейса и имеет следующие методы:

* метод state является публичным, не принимает входных параметров и отвечает за информацию о текущем состоянии отображения;
* метод draw является публичным, не принимает входных параметров и отвечает за корректную отрисовку;
* метод clear является публичным, не принимает входных параметров и отвечает за удаления текущего состояния отображения.

Класс «Set UP»представляет собой сущность, которая отвечает за персональные настройки пользователя и имеет следующие методы:

* метод changeTask является публичным, не принимает входных параметров и отвечает за изменения существующей задачи;
* метод changeLimitявляется публичным, не принимает входных параметров и отвечает за изменения сроков выполнения существующей задачи;

метод;

* метод changeWorker является публичным, не принимает входных параметров и отвечает за изменения сотрудника, который должен выполнить существующею задачу.

Класс «Worker»представляет собой сущность менеджера и имеет следующие методы:

* метод info является публичным, не принимает входных параметров и отвечает за передачу информации о сотруднике;
* метод add является публичным, не принимает входных параметров и отвечает за добавления нового сотрудника;
* метод delete является публичным, не принимает входных параметров и отвечает за удаления существующего сотрудника;
* метод clear является публичным, не принимает входных параметров и отвечает за отчистку состояния существующего сотрудника;
* метод state является приватным, не принимает входных параметров и отвечает за представления состояния сотрудника.

Так же класс «Worker» имеет следующие поля:

* поле name являться типом данных String, оно приватное и отвечает за персональную информацию сотрудника;
* поле address являться типом данных String, оно приватное и отвечает за информацию о месте жительства сотрудника.

Класс «Current result» представляет собой сущность статистики, которую менеджер может просмотреть, для принятия решения кому именно из сотрудников нужно планировать задачу и имеет следующие методы:

* метод day является публичным, принимает одни входных параметр типа String и отвечает за информацию о сотруднике за день;
* метод week является публичным, принимает одни входных параметр типа String и отвечает за информацию о сотруднике за неделю;
* метод month является публичным, принимает одни входных параметр типа String и отвечает за информацию о сотруднике за месяц;
* метод year является публичным, принимает одни входных параметр типа String и отвечает за информацию о сотруднике за год;
* метод statistick является публичным, принимает одни входных параметр типа String и отвечает за информацию о статистике.

Проектирования WEB части приложения для дистанционного управления сотрудниками будит реализовано при помощи языка программирования javascript и фреймворка React. Разметка страниц сайтов, будит сделано при помощи HTML5, CSS фреймворка Bootstrap.

Средой разработки будит являться текстовый редактор Visual Studio Code, так как он обладает всеми необходимыми свойствами и плагинами, которые необходимы при разработке. Так же он распространяется на все популярные операционные системы (Windows, Linux, MacOS).

3.3 Проектирование базы данных приложения

В мире существует множество систем управления базами данных. Для реализации базы данных системы для контроля персоналам необходимо, чтобы система управления базами данных (СУБД) поддерживала следующие возможности:

* базировалась на языке SQL;
* встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
* высокопроизводительные и надёжные механизмы транзакций;
* расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных);
* встроенный процедурный язык;
* поддержка одновременной модификации БД несколькими пользователями;
* поддержка набора встроенных типов данных;
* поддержка запросов с OUTER JOIN, UNION, UNION ALL, EXCEPT, INTERSECT и подзапросов;
* контроль целостности;
* поддержка регулярных выражений.

С этими требованиями соответсвует несколько СУБД, а именно SQLite, MySQL и PostgreSQL.

SQLite — это хорошо разработанная библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой (в сравнении с серверными БД) обработки любых видов данных [3]. Но аппаратное обеспечения, но котором будит происходить физическая реализация не подходит под данную СУБД.

MySQL — это самая популярная из всех крупных серверных БД. В её официальной документации можно найти большое количество информации. Хотя MySQL и не пытается полностью реализовать SQL-стандарты, она предлагает широкий функционал. Приложения общаются с базой данных через процесс-демон [3]. Так как данная СУБД полностью не реализует SQL-стандарты то она нам не подходить.

PostgreSQL - свободная объектно-реляционная СУБД. PostgreSQL базируется на языке SQL [3].

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это делает PostgreSQL гибким и надежным.

Исходя из выше перечисленных требований и просмотренных СУБД было принята решение сделать выбор в пользу PostgreSQL

В результате даталогического проектирования были созданы отношения (PERSON, WORKER, MANAGER, TASK, CHAT), связи между которыми отображены на даталогической модели данных, которая представлена на рисунке 11.

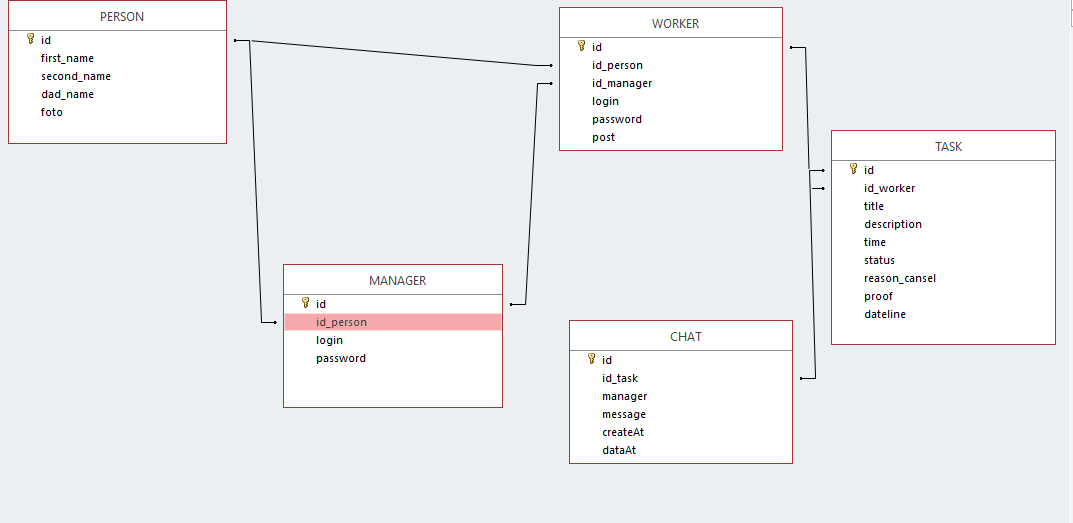


Рисунок 11 – Даталогическая модель данных

3.4 Описание таблиц базы данных

Рассмотрим таблицы созданной даталогической модели.

Таблица PERSON хранит личную информацию о отдельном человеке (менеджер, сотрудник), в которую входит имя, фамилия, отчество и URL ссылка на фото, таблица 1.

Таблица 1 – Таблица «PERSON»

|  |  |
| --- | --- |
| Имя атрибута | Домен атрибута |
| id (прим.: идентификатор) | SERIAL NOT NULL |
| first\_name (прим.: имя) | TEXT NOT NULL |
| second\_name (прим.: фамилия) | TEXT NOT NULL |
| dad\_name (прим.: отчество) | TEXT |
| foto (прим.: фото) | TEXT |

Ограничения:

id – первичный ключ (уникальные значения);

Таблица WORKER хранит информацию о сотруднике, его логин, пароль и должость, таблица 2.

Таблица 2 – Таблица «WORKER»

|  |  |
| --- | --- |
| Имя атрибута | Домен атрибута |
| id (прим.: идентификатор) | SERIAL NOT NULL |
| id\_person | INTENGER NOT NULL |
| id\_manager | INTENGER NOT NULL |
| login (прим.: логин) | TEXT NOT NULL |
| password (прим.: пароль) | TEXT NOT NULL |
| post (прим.: пост) | TEXT |

Ограничения:

id – первичный ключ (уникальные значения);

id\_person – внешний ключ к таблице «PERSON» поле «id», обновление каскадное, удаление каскадное;

id\_manager – внешний ключ к отношению «MANAGER» поле «id», обновление каскадное, удаление каскадное;

Таблица MANAGER хранит информацию о менеджере, его логин и пароль, таблица 3.

Таблица 3 – Таблица «MANAGER»

|  |  |
| --- | --- |
| Имя атрибута | Домен атрибута |
| id (прим.: идентификатор) | SERIAL NOT NULL |
| id\_person | INTENGER NOT NULL |
| login (прим.: логин) | TEXT NOT NULL |
| password (прим.: пароль) | TEXT NOT NULL |

Ограничения:

id – первичный ключ (уникальные значения);

id\_person – внешний ключ к таблице «PERSON» поле «id», обновление каскадное, удаление каскадное;

Таблица TASK хранит информацию о задачи, её заголовок, описание, время выполнения, причину отмены, статус, URL ссылку на фото отчет о выполненной работе, таблица 4.

Таблица 4 – Таблица «TASK»

|  |  |
| --- | --- |
| Имя атрибута | Домен атрибута |
| id (прим.: идентификатор) | SERIAL NOT NULL |
| id\_worker | INTENGER NOT NULL |
| title (прим.: заголовок) | TEXT NOT NULL |
| description (прим.: описание) | TEXT NOT NULL |
| time (прим.: время выполнения) | TEXT NOT NULL |
| status (прим.: статус задачи) | BOOLEAN |
| reason\_cansel(прим.: причина отмены) | TEXT |
| dateline(прим.: срок выполнения) | TEXT NOT NULL |
| proof (прим.: отчет) | TEXT NOT NULL |

Ограничения:

id – первичный ключ (уникальные значения);

id\_worker – внешний ключ к таблице «WORKER» поле «id», обновление каскадное, удаление каскадное;

Таблица CHAT хранит информацию чата задачи, его сообщение, тип ответа, время чтения и время отправления, таблица 5.

Таблица 5 – Таблица «CHAT»

|  |  |
| --- | --- |
| Имя атрибута | Домен атрибута |
| id (прим.: идентификатор) | SERIAL NOT NULL |
| id\_task | INTENGER NOT NULL |
| manager (прим.: менеджер) | BOOLEAN |
| message (прим.: сообщение) | TEXT NOT NULL |
| createAt (прим.: время создания) | TEXT NOT NULL |
| dataAt (прим.: время чтения) | TEXT NOT NULL |

Ограничения:

id – первичный ключ (уникальные значения);

id\_task – внешний ключ к таблице «TASK» поле «id», обновление каскадное, удаление каскадное;

# Заключение

В ходе выполнения выпускной преддипломной практики были решены все поставленные задачи.

Проведен обзор предметной области и выполнен сравнительный анализ систем управления персоналом.

Выполнено проектирование даталогической моделей системы.

Обоснован выбор программного инструментария: WEB часть система реализована на базе фреймворка React при помощи языка программирование Javascript, для управления базой данных выбрана СУБД PostgresSQL, разметка документов реализована на основе HTML, CSS, фреймворка Bootstrap, мобильная часть системы реализована на базе операционной системы Android при помощи языка программирования Kotlin и библиотеки Retrofit для обмена данными с сервером.

Проведено проектирование системы для дистанционного управления персоналом и разработана система, отвечающая всем требованиям технического задания.

# Перечень использованных информационных ресурсов

1. Брекнер А. Гибкое управление проектами с канбан.– Наука и техника, 2018.– 352 с.
2. Андреева И.О. Управление кадрами. Руководство для персонала и топ-менеджмента. – М.: НОИ Интуит, 2016. – 271 с.
3. Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2000. – 1120 с.
4. Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов.– Эксмо, 2017.– 480 с.
5. Котеров Д. PHP 7/ Д. Котеров, И. Симдянов.– СПб.: БХВ-Петербург, 2018.– 1088 с.
6. Зандстра М. PHP. Объекты, шаблоны и методики программирования.– Вильямс, 2016.– 576 с.
7. Поляков Е. PHP на примерах.– Наука и техника, 2017.– 256 с.
8. Маркин А. Основы Web-программирования на PHP/ А. Маркин, С. Шкарин.– Диалог-МИФИ, 2012.– 256 с.
9. Колисниченко Д. PHP и MySQL. Разработка Web-приложений.– СПб.: БХВ-Петербург, 2017.– 640 с.
10. Исаев Г. Проектирование информационных систем.– Омега-Л, 2012.– 432 с.
11. Грекул В. Проектирование информационных систем. Учебник и практикум/ В. Грекул, Н. Коровкина, Г. Левочкина.– Юрайт, 2018.– 376 с.
12. Васильев А. JavaScript в примерах и задачах.– Эксмо, 2017.– 720 с.
13. Браун Э. Изучаем JavaScript. Руководство по созданию современных веб-сайтов.– Альфа-книга, 2017.–368 с.
14. Веллинг Л. Разработка веб-приложений с помощью PHP и MySQL/Л.Веллинг, Л. Томсон.–Вильямс, 2017.– 768 с.

Приложение А Техническое задание

|  |  |
| --- | --- |
| СОГЛАСОВАНО  ст. преп. кафедры «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ А.П. Кузин  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. | УТВЕРЖДЕНО  Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ В.В. Долгов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

А.1. Общие сведения

### А.1.1 Имя программы

«KiWorker»

### А.1.2 Полное наименование системы

Система для дистанционного управления персоналом «KiWorker»

### А.1.3 Перечень документов, на основании которых создается система

Основанием для разработки является задание к выпускной квалификационной работе (ВКР), согласованное с руководителем ВКР старшим преподавателем кафедры «ПОВТиАС» Кузиным Александром Павловичем с одной стороны, студентом гр. ВПР-41 Русановым Семёном Алексеевичем, именуемым в дальнейшем исполнителем с другой стороны, утвержденному заведующим кафедрой «ПОВТиАС» Долгову Василию Валерьевичу.

### А.1.4 Порядок оформления и предъявления заказчику результатов работ по созданию системы

Система передается в виде функционирующего комплекса на базе средств вычислительной техники Заказчика и Исполнителя. Приемка системы осуществляется комиссией в составе уполномоченных представителей Заказчика и Исполнителя.

### А.1.5 Перечень нормативно–технических документов, методических материалов, использованных при разработке ТЗ

При разработке автоматизированной системы и создании проектно–эксплуатационной документации Исполнитель должен руководствоваться требованиями следующих нормативных документов:

ГОСТ 19.201–78. Техническое задание. Требования к содержанию и оформлению.

ГОСТ 34.601–90. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

ГОСТ 34.201–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем;

РД 50–34.698–90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

### А.2 Назначение системы для управления персоналом.

Основным назначением системы автоматизация бизнес-процессов компании, при помощи дистанционного управления персоналом.

### А.3 Требования к системе для управления персоналом

### А.3.1 Требования к системе для управления персоналом в целом

### А.3.1.1 Требования к графическому дизайну системы для управления персоналом

Дизайн системы должен соответствовать дизайну, утвержденному Заказчиком

### А.3.1.2 Требования к шрифтовому оформлению системы для управления персоналом

Основным шрифтом должен стать Times New Roman*.*

Размер (кегль) шрифтов должен обеспечивать удобство восприятия текста при минимально допустимом размере экрана.

### А.3.1.3 Требования к средствам просмотра Сайта

Сайт должен обеспечивать корректное отображение данных в следующих браузерах:

* Microsoft Edge (версия 11);
* Opera (версия 76);
* Google Chrome (версия 90);
* Firefox Browser (версия 88).

### А.3.1.4 Требования к контенту и наполнению системы для управления персоналом

Первичная разработка и верстка контента (информационного содержимого) Сайта должна производиться силами Исполнителя при согласовании с Заказчиком. Заказчик предоставляет все необходимые Исполнителю текстовые и графические материалы, а также комментарии, касающиеся их содержания, объема, оформления и размещения.

### А.3.1.5 Требования к компоновке страниц системы для управления персоналом

Компоновка страниц системы должна обеспечивать минимальный размер страниц в зависимости от ширины рабочего поля браузера пользователя. Минимальный размер (ширина) рабочего поля браузера, при котором необходимо обеспечить полноценное отображение страниц (без полосы горизонтальной прокрутки), составляет 1024 пиксела.

### А.3.1.6 Язык сайта

Русский.

### А.3.2 Требования к функциям (возможностям), выполняемых сайтом

Сайт должен давать возможность: планирования задач, просмотр статистики, как общей, так и отдельного сотрудника, переноса невыполненных задач на следующий день

### А.3.2.1 Основные требования

Система должна представлять собой веб-сайт, размещенный в сети Интернет, а также мобильное приложения на ОС Android

### A.4 Требования к программе или программному изделию

### А.4.1 Требования к функциональным характеристикам

Программное средство должно осуществлять следующие функции:

* возможность планирования задач менеджером для конкретного сотрудника;
* просмотр менеджером статистики выполнения задач;
* добавления менеджером нового сотрудника в систему;
* удаления менеджером сотрудника из системы;
* изменения данных менеджером о сотруднике в системе;
* просмотр своего списка задач сотрудником;
* возможность сотрудником отметить задачу как выполненную;
* возможность прикрепления отчета о выполненной задачи;
* возможность отмены задачи сотрудником;
* уточнения деталей выполнения задачи.

### А.4.2 Требования к надежности

Надежное функционирование программы должно быть обеспечено выполнением совокупности нижеописанных мероприятий:

* выполнение требований ГОСТ 51188-98;
* защита информации;
* организацией бесперебойного питания технических средств;
* использованием лицензионного программного обеспечения.

### А.4.3 Условия эксплуатации

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации мобильной техники.

Высокая квалификация пользователя программного средства не требуется. Дополнительных требований и ограничений не вводится.

Требования к персоналу, работающему с данным программным продуктом – общие знания вычислительной техники.

### А.4.4 Требования к составу и параметрам технических средств

Состав технических средств на стороне клиента: мобильное устройство с камерой.

### А.4.5 Требования к информационной и программной совместимости

Для функционирования программного средства на стороне клиента необходимо следующее программное обеспечение: ОС Android версии 10 и старше.

### А.4.5.1 Язык программирования

Для разработки программного средства должен быть использован язык программирования Kotlin.

### А.4.5.2 Операционная система

Программное средство должно работать под управлением ОС Android.

### А.4.6 Требования к упаковке и маркировке

Требования к упаковке и маркировке программного средства не предъявляется.

### А.4.7 Требования к транспортировке и хранению

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное программное изделие.

Допустимы все способы транспортирования и хранения, не нарушающие целостность используемого носителя данных. Программное средство может храниться на любом носителе информации, имеющее возможность подключения к персональному компьютеру.

### А.4.8 Специальные требования

Для корректной работы программного средства необходимо разрешение на работу с памятью устройства, чтобы обеспечить корректное открытие и сохранение файлов, и разрешение на работу с камерой, чтобы обеспечить быстрое получение нового изображения.

### А.5 Требования к программной документации

Программная документация должна состоять из следующих листов:

* титульный лист;
* пояснительная записка к производственной практике;
* техническое задание по ГОСТ 19.201-78 ЕСПД;
* исходный код программного средства по ГОСТ 19.401-79 ЕСПД.

### А.6 Стадии и этапы разработки

* постановка задачи (с 20.04.20 по 23.04.20);
* изучение предметной области (с 23.04.20 по 26.04.20);
* разработка алгоритмов решения задачи (с 27.04.20 по 02.05.20);
* разработка программы (с 04.05.20 по 09.05.20);
* тестирование программы (с 11.05.20 по 14.05.20).

### А.7 Порядок контроля и приемки

Порядок и контроль приёмки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие правильно работающего программного средства с тестовым примером и отчета, представленного в печатном виде.

Разработчик технического задания /Русанов С.А. /

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпис

# Приложение Б

# Листинг программы

Листинг Б.1 – Основная активность приложения

from kivy.core.window import Window

from kivy.app import App

import shutil

import os

from kivy.uix.anchorlayout import AnchorLayout

from kivy.uix.gridlayout import GridLayout

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.textinput import TextInput

from kivy.uix.button import Button

from kivy.uix.switch import Switch

from kivy.uix.widget import Widget

from kivy.uix.image import Image

from kivy.uix.label import Label

from GoogleImages import GoogleImages

from Database import Database

from TextNN import TextNN

from Manager import Manager

from Image import Image as ImageUser

class MyApp(App):

textInput = TextInput(size\_hint=(1, .18), halign='center', font\_size=16)

switch = Switch(size\_hint=(.3, .3), active=True)

switchText = Switch(size\_hint=(.1, .1), active=True)

img = Image(source="1.jpg")

buttonSearch = Button(text='Start', size\_hint=(.3, .3))

switchValue = True

images = []

index = 1

path = ""