МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»  
(Самарский университет)

Институт информатики и кибернетики

Кафедра программных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**«РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**УПРАВЛЕНИЯ ИТ-ПРОЕКТАМИ»**

по направлению подготовки 02.03.02

Фундаментальная информатика и информационные технологии

(уровень бакалавриата)

направленность (профиль) «Информационные технологии»

Обучающийся Г.С. Вазюкова

(подпись, дата)

Руководитель ВКР

к.т.н., доцент Е.В. Сопченко

(подпись, дата)

Нормоконтроллер Е.В. Сопченко

(подпись, дата)

Самара 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»  
(Самарский университет)

Институт информатики и кибернетики

Кафедра программных систем

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.В. Востокин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ   
(бакалавр)

обучающейся Вазюковой Галине Сергеевне

группа 6413-020302D

Тема работы: Разработка информационной системы управления

ИТ-проектами

Исходные данные см. в приложении к заданию

Структурные части работы (перечень вопросов, подлежащих разработке):

1. Провести анализ предметной области: управление проектами, методологии Agile, процесс командной разработки
2. Сделать обзор систем-аналогов в области управления ИТ-проектами, реализующих методологии Agile
3. Разработать структурную схему системы
4. Разработать информационно-логический проект системы по методологии UML
5. Разработать интерфейс пользователя
6. Разработать и реализовать программное и информационное обеспечение, провести   
   тестирование и отладку
7. Провести ресурсные расчеты

|  |  |
| --- | --- |
| Научныйруководитель  доцент, кафедра программных систем | Задание принял к исполнению |
| (E.В. Сопченко)  « 26 » октября 2021 г. | (Г.С. Вазюкова)  « 26 » октября 2021 г. |

ПРИЛОЖЕНИЕ

к заданию на выпускную квалификационную работу бакалавра

обучающейся Г.С. Вазюковой группы № 6413-020302D

Тема: «Разработка информационно системы управления ИТ-проектами»

Исходные данные к работе:

1. Характеристики объекта автоматизации:
2. объект автоматизации – процесс управления ИТ-проектами;
3. виды автоматизируемой деятельности:

* процесс создания проекта;
* процесс изменения конфигурации проекта;
* процесс выдачи ролей участников проекта;
* процесс создания задач;
* процесс распределения задач;
* процесс визуализации доски размещения задач;

1. Требования к информационному обеспечению:
2. Информационное обеспечение разрабатывается на основании следующих литературных источников:

* Плюсы и минусы командной работы [Электронный ресурс]. URL: https://4brain.ru/blog/%D0%BF%D0%BB%D1%8E%D1%81%D1%8B-%D0%B8-%D0%BC%D0%B8%D0%BD%D1%83%D1%81%D1%8B-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B/ (дата обращения 24.12.2021).
* Методологии управления проектами [Электронный ресурс]. URL: https://www.wrike.com/ru/project-management-guide/metodologii-upravlenija-proektami/ (дата обращения: 24.12.2021).

1. Структура базы данных разрабатывается на основании следующих сведений:

* о задачах (название, исполнитель, приоритет, тип, состояние, статус принятия, описание);
* о проектах (название, менеджер, схема типов задач, схема рабочего процесса, список участников);
* о пользователях (логин, пароль, имя, роль);
* о схемах типов задач (название, список типов);
* о типах задач (название);
* о схемах рабочего процесса (название, список состояний);
* о состояниях (название).

1. Обеспечить контроль целостности базы данных.
2. Требования к техническому обеспечению:
   1. Требования к техническому обеспечению серверной части:
3. тип ЭВМ: x86-64 совместимый;
4. объем ОЗУ – не менее 3 Гб;
5. объем свободного дискового пространства – не менее 50 Гб;
6. клавиатура или иное устройство ввода;
7. мышь или иное манипулирующее устройство;
8. процессор – Intel Core i3 не менее 1,5 ГГц;
9. дисплей с разрешением не менее 1024 × 768 пикселей;
10. операционная система Windows 7 и выше;
11. PostgreSQL 11 и выше.
    1. Требования к техническому обеспечению клиентской части:
12. тип ЭВМ: ARM совместимый;
13. объем ОЗУ – не менее 3 Гб;
14. процессор – с частотой не менее 1,5 ГГц;
15. дисплей с разрешением не менее 800 × 600 пикселей;
16. операционная система Android 5.0 или выше (iOS 5.0 или выше).
17. Требования к программному обеспечению:
    1. Требования к программному обеспечению серверной части:
18. тип операционной системы – Windows 7 и выше;
19. СУБД – PostgreSQL 10;
    1. Требования к программному обеспечению клиентской части:
20. тип операционной системы – Windows 7 и выше, Android;
21. браузер – Google Chrome 86.0.4240.183 (64-битный) и выше, Firefox 83.0 (64-битный) и выше.
    1. Требования к программному обеспечению рабочего места разработчика:
22. тип операционной системы – Windows 7 и выше;
23. язык программирования – Java, JavaScript;
24. среда программирования – IntelliJ IDEA 2019;
25. СУБД – PostgreSQL 10;
26. среда проектирования – StarUML 4.1.0.
27. Общие требования к проектируемой системе.
    1. Функции, реализуемые системой:
28. Общесистемные функции:

* ведение базы данных;
* авторизация пользователя;

1. Функции системного администратора:

* работа с проектами:
* работа с типами задач:
* работа с рабочим процессом:
* работа с пользователями;
* работа со службой технической поддержки;

1. . Функции менеджера проекта:

* создание задачи;
* добавление участников проекта;

1. Функции участника проекта:

* перевод задачи по состояниям.
  1. Технические требования к системе:

1. режим работы – диалоговый;
2. температура окружающего воздуха – 15-25°С;
3. влажность окружающего воздуха – 45-75%;
4. система должна удовлетворять санитарным правилам и нормам СанПин 2.2.2/2.4.2198-07;
5. условия работы средств вычислительной техники должны соответствовать ГОСТ 12.1.005, 12.1.007.

Научный руководитель,

к.т.н., доцент Е.В. Сопченко

подпись, дата

РЕФЕРАТ

Пояснительная записка: 84 с, 39 рисунков, 12 таблиц, 24 источника, 2 приложения.

Графическая часть: 12 слайдов презентации PowerPoint.

AGILE, KANBAN, МЕНЕДЖЕР, УПРАВЛЕНИЕ ПРОЕКТОМ, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ЗАДАЧА

Объектом исследования является управление командной работов в ИТ-проектах.

Цель работы – разработать информационную систему управления ИТ-проектами.

В процессе работы были разработаны алгоритмы и соответствующая программа, позволяющая пользователю настроить управление работой проекта по методологии Agile. Система позволяет создавать

Серверная часть системы разработана на языке Java с использованием фреймворка Spring Boot и функционирует под управлением операционных систем Windows 7/8/10. Доступ к данным осуществляется с помощью СУБД PostgreSQL 11.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

|  |  |  |
| --- | --- | --- |
| API | − | Application programming interface (программный интерфейс приложения) |
| REST | − | Representational State Transfer (передача репрезентативного состояния) |
| БД | − | база данных; |
| ПО | − | программное обеспечение; |
| СУБД | − | система управления базами данных; |

Содержание

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 8](#_Toc104646068)

[Введение 11](#_Toc104646069)

[1 Описание и анализ предметной области 12](#_Toc104646070)

[1.1 Работа в команде 12](#_Toc104646071)

[1.2 Методологии управления проектами 12](#_Toc104646072)

[1.2.1 Каскадная методология управления проектами (Waterfall) 12](#_Toc104646073)

[1.2.2 Методология управления проектами Agile 14](#_Toc104646074)

[1.2.2.1. Scrum 14](#_Toc104646075)

[1.2.2.2. Kanban 15](#_Toc104646076)

[1.3 Обзор существующих систем-аналогов 17](#_Toc104646077)

[1.3.3 JIRA 17](#_Toc104646078)

[1.3.4 Trello 18](#_Toc104646079)

[1.4 Диаграмма объектов предметной области 21](#_Toc104646080)

[1.5 Постановка задачи 21](#_Toc104646081)

[1.5.1 Режим системного администратора 22](#_Toc104646082)

[1.5.2 Режим менеджера проекта 23](#_Toc104646083)

[1.5.3 Режим участника проекта 23](#_Toc104646084)

[2 Проектирование системы 24](#_Toc104646085)

[2.1 Выбор и обоснование архитектуры системы 24](#_Toc104646086)

[2.2 Структурная схема системы 24](#_Toc104646087)

[2.3 Разработка прототипов экранных форм приложения 26](#_Toc104646088)

[2.3.1 Мобильное приложение 26](#_Toc104646089)

[2.3.2 Веб-приложение 29](#_Toc104646090)

[2.4 Разработка информационно-логического проекта системы 30](#_Toc104646091)

[2.4.3 Язык UML 30](#_Toc104646092)

[2.4.4 Диаграмма вариантов использования 31](#_Toc104646093)

[2.4.5 Диаграмма классов 32](#_Toc104646094)

[2.4.6 Диаграмма деятельности 33](#_Toc104646095)

[2.4.7 Логическая модель данных системы 37](#_Toc104646096)

[2.5 Выбор и обоснование комплекса программных средств 43](#_Toc104646097)

[2.5.1 Выбор языка программирования 44](#_Toc104646098)

[2.5.2 Выбор операционной системы 45](#_Toc104646099)

[2.5.3 Выбор среды программирования 45](#_Toc104646100)

[2.5.4 Выбор СУБД 47](#_Toc104646101)

[3 Реализация системы 48](#_Toc104646102)

[3.1 Разработка и описание интерфейса пользователя 48](#_Toc104646103)

[3.1.1 Экранные формы мобильного приложения 48](#_Toc104646104)

[3.1.2 Экранные формы веб-приложения 52](#_Toc104646105)

[3.2 Диаграммы реализации 58](#_Toc104646106)

[3.2.1 Диаграмма развертывания 58](#_Toc104646107)

[3.2.2 Диаграмма компонентов 58](#_Toc104646108)

[3.2.3 Диаграмма классов 59](#_Toc104646109)

[3.3 Выбор и обоснование комплекса технических средств 59](#_Toc104646110)

[3.3.1 Расчет объема занимаемой памяти. Клиентская часть, использующая мобильное приложение 59](#_Toc104646111)

[3.3.2 Расчет объема занимаемой памяти. Клиентская часть, использующая веб-приложение 62](#_Toc104646112)

[3.3.3 Расчет объема занимаемой памяти. Серверная часть 63](#_Toc104646113)

[Заключение 66](#_Toc104646114)

[Список использованных источников 67](#_Toc104646115)

Введение

В наше время разработка и ведение проектов часто осуществляется в команде. Для грамотной реализации командной работы проектные менеджеры часто прибегают к использованию методологии организации работы в команде. Методологии управления проектами представляют собой разные подходы к организации проекта. Следуя четким правилам по организации разработки, команда может:

* четко поставить цель и правильно ее достичь;
* корректно распределить задачи между участниками проекта;
* выделять более приоритетные задачи для выполнения;
* следить за продвижением проекта в целом.

Наиболее удобным способом использования методологий являются специализированные системы, которые помогают избавиться от ведения дел на бумаге. При помощи этих приложений работа в команде упрощается не только для руководителя, но и для участников проекта, т.к. каждому из них четко понятны все задачи и требования, которые необходимо выполнить.

Также в настоящее время приобретают большую популярность мобильные приложения по ряду причин:

* Отсутствие необходимости иметь персональный компьютер для доступа к приложению;
* Удобство использования – получить необходимый функционал можно в любое время.

В ходе выполнения выпускной квалификационной работы необходимо разработать мобильное приложение для организации и ведения работы в команде. А также веб-приложение для организации системного администрирования.

1. Описание и анализ предметной области
   1. Работа в команде

Разработка проектов зачастую ведется в команде. На это есть несколько причин [1]:

* разработка проекта осуществляется гораздо быстрее;
* командная разработка представляет собой одну из форм делегирования полномочий.
* командная работа наделяет людей способностью к сотрудничеству.

Чтобы работа команды была успешной, необходимо соблюдать следующие условия:

* грамотная постановка целей;
* наличие четких и ясных задач;
* правильный подбор командного состава;
* наличие детализированной системы работы.

Системы планирования, постановки и контроля задач — основной инструмент работы над проектами, который поможет в выполнении этих условий.

* 1. Методологии управления проектами
     1. Каскадная методология управления проектами (Waterfall)

Самый распространенный способ планирования проектов – составление последовательности задач, приводящей к конечному результату, и поочередное выполнение этих задач. Так выглядит каскадная методология – традиционный и наиболее простой для понимания метод управления проектами. Предыдущая задача должна быть выполнена до начала выполнения следующей задачи в составе связной последовательности элементов, в совокупности приводящих к конечному результату [2].

Достоинства [3]:

* Понятная и простая структура процесса разработки – это снижает порог вхождения для команд.
* Удобная отчётность – можно легко отследить ресурсы, риски, затраченное время и финансы благодаря строгой этапности процесса разработки и детальной документации проекта.
* Стабильность задач – задачи, которые стоят перед продуктом, ясны команде с самого начала разработки, и остаются неизменными на протяжении всего процесса.
* Оценка стоимости и сроков сдачи проекта – сроки выпуска готового продукта, как и его итоговая стоимость могут быть просчитаны до момента запуска разработки.

Недостатки [3]:

* Лишенный гибкости процесс – так, если проект требует больше временных и финансовых ресурсов, чем возможно, то команда может пожертвовать фазой тестирования. Согласно исследованиям консалт-группы Rothman, стоимость исправления дефектов после выпуска продукта выше в среднем в 20 раз, чем во время полноценного многоэтапного тестирования в процессе разработки.
* «Стойкость» к изменениям – жёсткий каркас из этапов разработки и условие предоставление только готового продукта определяют невозможность вносить изменения во время разработки.
* Инерционность – на первых стадиях прогноз временных и финансовых трат может измениться в сторону увеличения, но изменить проект в сторону оптимизации затрат, изменения функционала или концепции до выпуска готового продукта невозможно.
* Повышенный риск – классическая система тестирования подразумевает отдельно тестирование каждого из компонентов проекта, в том числе, во взаимодействии с другими. При использовании Waterfall происходит тестирование готового продукта.
  + 1. Методология управления проектами Agile

Ключевые принципы методологии Agile были разработаны в 2001 году и включают четыре основных ценности [2]:

* люди и взаимодействие важнее процессов и инструментов;
* работающий продукт важнее исчерпывающей документации;
* сотрудничество с клиентом важнее согласования условий контракта;
* готовность к изменениям важнее следования первоначальному плану.
  + - 1. Scrum

Scrum — наиболее популярный фреймворк методологии Agile за счет относительной простоты реализации. Он также позволяет решить множество проблем, с которыми ранее сталкивались разработчики программного обеспечения: например, запутанность циклов разработки, отсутствие гибкости планов проектов и сдвиги производственных графиков.

В методологии Scrum работу небольшой команды организует скрам-мастер, чья основная задача — устранить все препятствия на пути более эффективного выполнения работы. Рабочий процесс команды делится на короткие двухнедельные циклы, называемые «спринты», при этом участники команды ежедневно обсуждают проделанную работу и помехи, которые необходимо устранить. Эта методология ускоряет разработку и тестирование, особенно в небольших командах [2].

Достоинства [4]:

* команда работает короткими этапами, на каждом из которых определяет цели и пути их достижения, что ускоряет процесс работы;
* команда работает над разными задачами проекта одновременно, что позволяет быстрее достичь желаемой цели;
* большие задачи разделяют на мелкие, поэтому внести корректировки прямо в процессе работы намного проще, чем в каскадном подходе;
* сокращается время на поиск ошибок и объяснение проблем;
* присутствует открытый обмен информацией, что делает процесс работы максимально прозрачным;
* поддержание высокого уровня мотивации в команде благодаря ежедневной видимости достижений.

Недостатки [4]:

* успех проекта во многом зависит от скрам-мастера (организатор процесса), квалификации команды и их приверженности своему делу;
* далеко не всегда можно адаптировать метод скрам под сферу деятельности, поскольку есть проекты, требующие исключительно планового подхода в работе;
* требует регулярной коммуникации с заказчиком, что порой тормозит процесс из-за невозможности получения обратной связи;
* сложность внедрения в масштабных и сложных проектах, так как больше подходит для малых и средних.
  + - 1. Kanban

Kanban — еще один фреймворк внедрения Agile-методологий, основанный на уровне загрузки команды. Этот метод был разработан на заводах компании Toyota в 1940-х годах и изначально представлял собой визуальную систему карточек («канбанов»), используемых отделами в качестве сигналов о готовности команды к обработке следующей партии сырья, о способности команды производить больше.

В наше время этот визуальный подход к управлению проектом отлично подходит для работы, требующей постоянной отдачи. Проектные группы создают визуализацию своих задач, зачастую при помощи стикеров и досок и двигают стикеры с задачами от первой до последней заранее установленной стадии, чтобы отслеживать прогресс в ходе выполнения задач и выявлять препятствия [2].

На рисунке 1 представлен пример реализации Kanban-доски. Все проекты в ней разделяются по статусу разработки. Также присутствует список потенциальных проектов. Методика планирования подчиняется ряду правил:

1. списки состояний первоначально пусты, заполнен лишь список потенциальных проектов;
2. проекты, которые необходимо взять в разработку перемещаются в первое состояние – начала разработки;
3. в ходе разработки проекта он перемещается в следующий по порядку столбец доски при достижении определенного состояния;
4. при завершении работы проект пропадает с доски и его статус становится «Закрыт».

  
Рисунок 1 – Пример реализации Kanban-доски

Достоинства [5]:

* гибкое планирование с возможностью изменения приоритетов — важно для проектов в IT-сфере;
* высокую степень вовлеченности сотрудников, благодаря совместным обсуждениям задач;
* экономию пространства на складах из-за сбалансированного потока транспортировки товаров;
* прозрачность рабочих процессов.

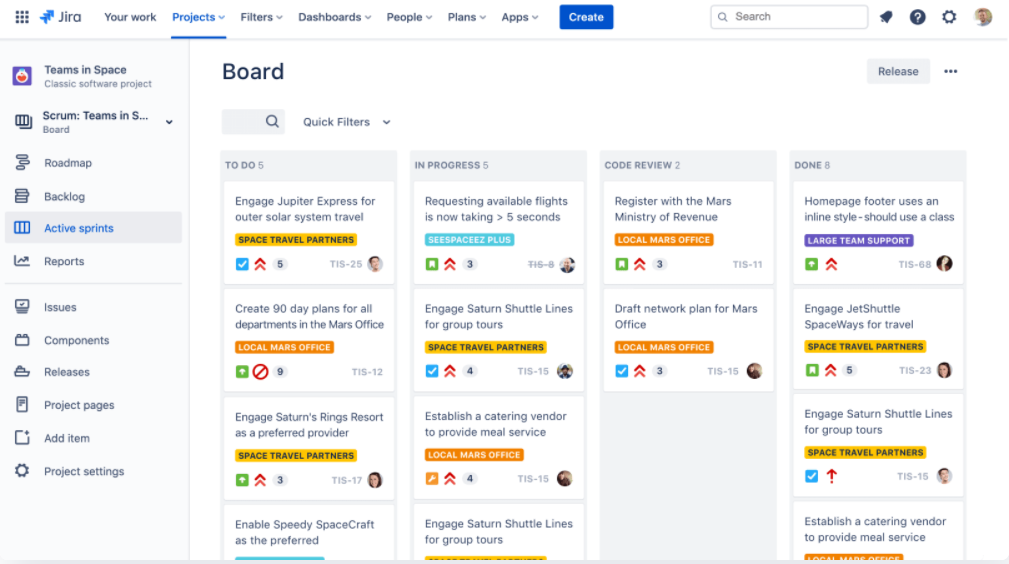
Недостатки [5]:

* неудобен для больших команд;
* невозможность долгосрочного планирования, так как Канбан предназначен для решения актуальных задач;
* вероятность торможения проекта при блокировке нескольких задач.
  1. Обзор существующих систем-аналогов

В настоящее время реализовано множество приложений для организации работы в команде, использующей методологию Kanban. Их анализ поможет подчеркнуть важные детали, на которые нужно обратить внимание при реализации системы.

* + 1. JIRA

Atlassian Jira Software – профессиональный инструмент разработки и управления проектами для agile-команд. Решение позволяет создавать пользовательские истории и задачи, планировать спринты и распределять задания в своей команде разработчиков. Можно использовать стандартный процесс или создавать свой собственный, подходящий именно конкретной команде[6]. На рисунке 2 приведена главная экранная форма программы «JIRA», на которой представлен пример реализации доски по методологии Kanban.

  
Рисунок 2 – Экранная форма программы «JIRA»

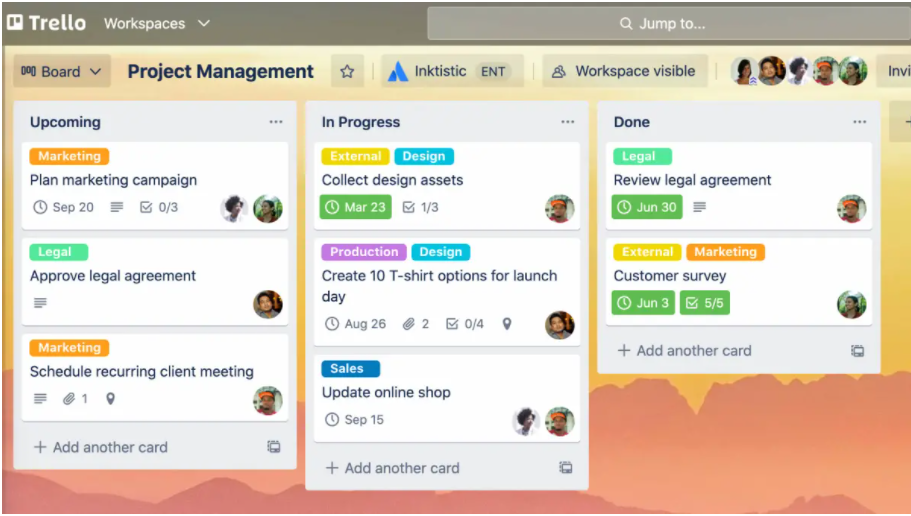
К достоинствам данной системы относятся:

* возможность реализации собственной методологии управления проектом;
* возможность интеграции с другими сервисами и отслеживания работы – системы code-review, документации и т.д.;
* возможность просматривать отчеты по итогам работы команды;
* возможность кастомизации при помощи загрузки сторонних плагинов.

К недостаткам системы относятся:

* сложность в технической поддержке;
* тяжеловесность;
* доступ к использованию функционала по платной подписке;
* отсутствие системы уведомлений без установки дополнительного ПО;
* неудобство работы в мобильной версии.
  + 1. Trello

Trello — это одна из самых популярных систем управления проектами в режиме онлайн, которая пользуется особенным спросом среди небольших компаний и стартапов. Она позволяет эффективно организовывать работу по японской методологии Kanban-досок[7]. На рисунке 3 приведена главная экранная форма программы «Trello», на которой приведен пример реализации Kanban-доски.

  
Рисунок 3 – Экранная форма программы «Trello»

К достоинствам данной системы относятся:

* простой интерфейс;
* почти неограниченный бесплатный доступ;
* возможность интеграции с другими популярными инструментами для онлайн-работы;

К недостаткам системы относятся:

* интерфейс становится неудобным при ведении нескольких проектов сразу;
* нет возможности добавлять описание проектов и модифицировать подзадачи;
* отсутствие системы уведомлений;
* неудобство работы в мобильной версии.

В таблице 1 приведена сравнительная характеристика рассмотренных систем по нескольким параметрам.

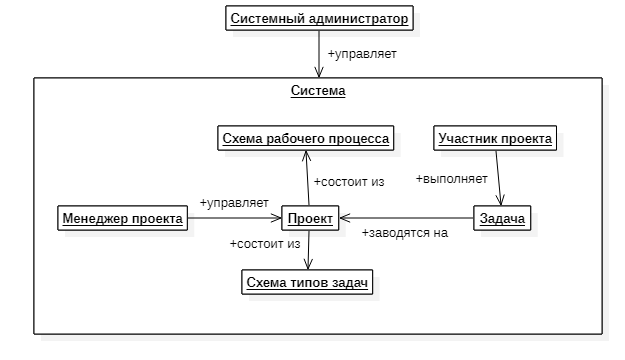
Таблица 1 – Сравнительная характеристика функциональности систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название показателя | JIRA | Trello | Разрабатываемая система |
| Удобство использования на мобильном устройстве | Нет | Нет | Да |
| Доступность системы уведомлений без установки дополнительного ПО | Нет | Нет | Да |
| Простота в освоении | Нет | Да | Да |
| Легковесность | Нет | Да | Да |
| Простота интерфейса при ведении нескольких проектов | Да | Нет | Да |
| Возможность модификации задач | Да | Нет | Да |
| Возможность добавления описания проекта | Да | Нет | Да |
| Возможность выдачи различных прав доступа к проекту | Да | Да | Да |

* 2. Диаграмма объектов предметной области

В предметной области системы управления проектами можно выделить такие основные объекты, как проект, задачи, которая создается на проекте, схема рабочего процесса, схема типов задач – из которых создается конфигурация проекта, участник проекта, который выполняет задачи, менеджер, который управляет проектом и системный администратор, который управляет системой.

На рисунке 4 приведена диаграмма объектов предметной области. Далее описать основные характеристики объектов.

  
Рисунок 4 – Диаграмма объектов предметной области

* 1. Постановка задачи

В результате выполнения выпускной квалификационной работы необходимо разработать мобильное приложение, с помощью которого можно управлять работой в команде: создавать и редактировать задачи в проектах, отслеживать процесс выполнения задач и работы каждого из участников проекта. А также веб-приложение для системного администрирования – создание и редактирование проектов, схем, а также их элементов – типов и состояний.

В системе должно быть реализовано три роли: системный администратор, проектный менеджер и участник проекта. Для каждой роли будет предусмотрена авторизация с использованием логина (длина от 6 до 12 символов), пароля (длина от 8 до 16 символов) и электронной почты.

Данные о пользователях, проектах, схемах проектов и задачах будет храниться в базе данных.

* + 1. Режим системного администратора

Системный администратор – пользователь, наделенный наибольшим количеством прав доступа к приложению.

Основные задачи системного администратора:

* создание и настройка конфигурации проектов;
* создание схем проектов;
* выдача прав для других пользователей приложения.

Конфигурация проекта настраивается на основе схем, которые создает системный администратор. В системе будет использоваться 2 вида схем для настройки проекта:

* схема рабочего процесса – набор статусов, по которым будет перемещаться задача;
* схема типов задач – набор типов задач, которые будут доступны для создания в проекте.

На проекте должны быть настроены две схемы разных типов (схема рабочего процесса и схема типов задач).

Также в приложении будет реализована подсистема обращения к системному администратору других пользователей. Она будет реализована в виде отдельного проекта, в который смогут заводить задачи все пользователи в независимости от роли. Проект будет настроен по особенным схемам:

* схема рабочего процесса будет иметь всего два статуса – задача открыта и выполнена;
* схема типов включает в себя один тип – обращение.
  + 1. Режим менеджера проекта

Проектный менеджер – пользователь, который будет управлять конкретным проектом. Он будет иметь возможность редактирования проекта, на который он назначен, при этом к другим проектам у него будет доступ только для просмотра.

Основные задачи проектного менеджера:

* создание новых задач внутри проекта;
* назначение задачи на участника проекта;
* добавление участников в проект
  + 1. Режим участника проекта

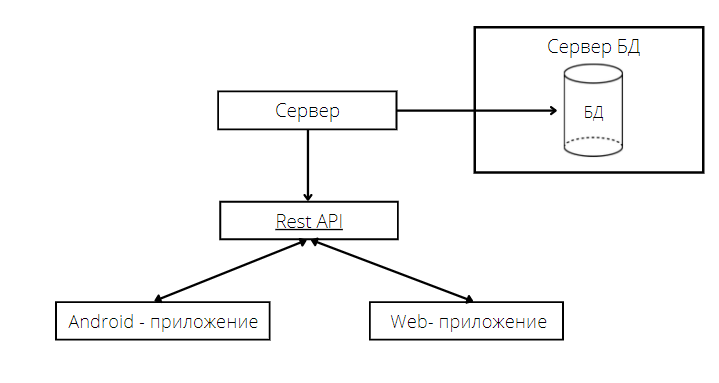
Участник проекта – пользователь, наделенный наименьшим количеством прав для управления системой. Основная задача участника – перемещение задачи из одного состояния в другое.

1. Проектирование системы
   1. Выбор и обоснование архитектуры системы

Архитектура системы — принципиальная организация системы, воплощенная в её элементах, их взаимоотношениях друг с другом и со средой, а также принципы, направляющие её проектирование и эволюцию [8].

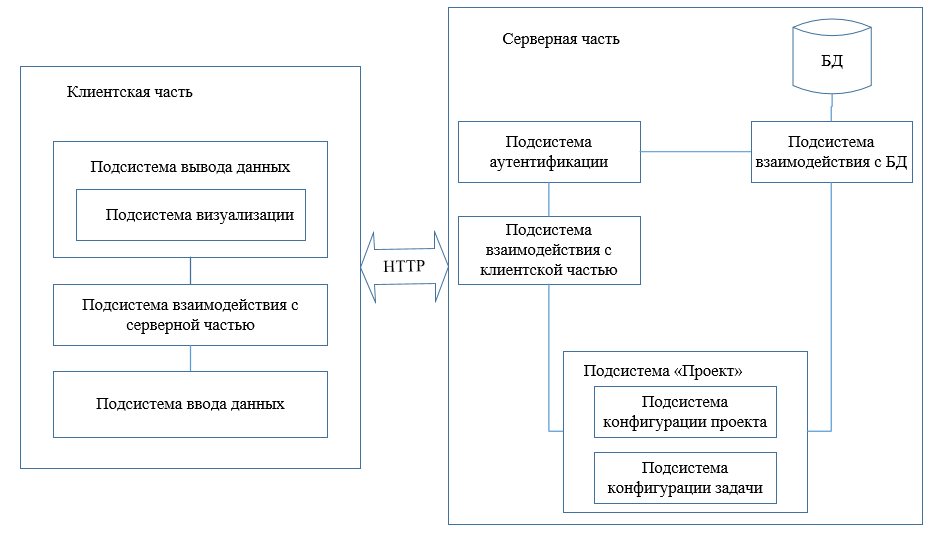
Для реализации приложения была выбрана клиент-серверная трехзвенная архитектура с двумя видами клиентов – мобильное приложение для менеджеров и участников и веб-приложение для системных администраторов. Логика обработки данных будет располагаться на сервере, который будет иметь REST API (интерфейс для взаимодействия клиентов и сервером). Данные в свою очередь будут храниться в базе данных, которая расположена на сервере БД.

На рисунке 5 приведена структурная схема реализации архитектуры разрабатываемой системы.

  
Рисунок 5 – Структурная схема реализации архитектуры системы

* 1. Структурная схема системы

На рисунке 6 приведена структурная схема разрабатываемой системы. Система разделена на две части (клиентская и серверная части, которые обмениваются информацией через HTTP-протокол).

  
Рисунок 6 – Структурная схема системы

В состав клиентской части входят следующие подсистемы:

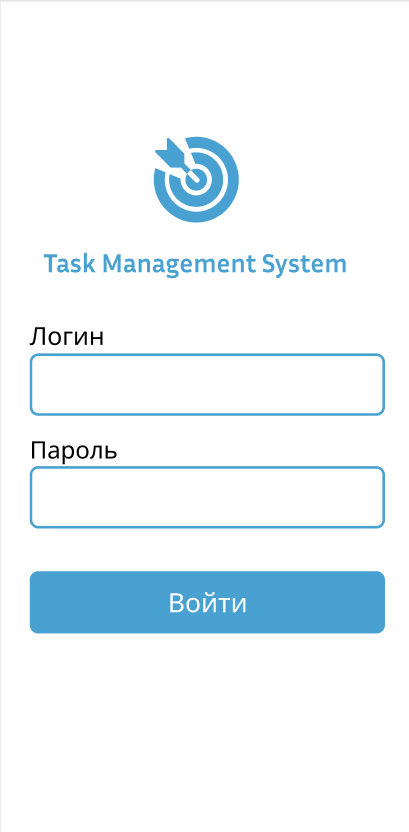
1. подсистема вывода данных, в состав которой входит подсистема визуализации, которая отвечает за вывод информации на экранную форму приложения;
2. подсистема ввода данных, которая отвечает за ввод информации с форм приложения;
3. подсистема взаимодействия с серверной частью, которая отвечает за взаимодействие с серверной частью.

В состав серверной части приложения входят следующие подсистемы:

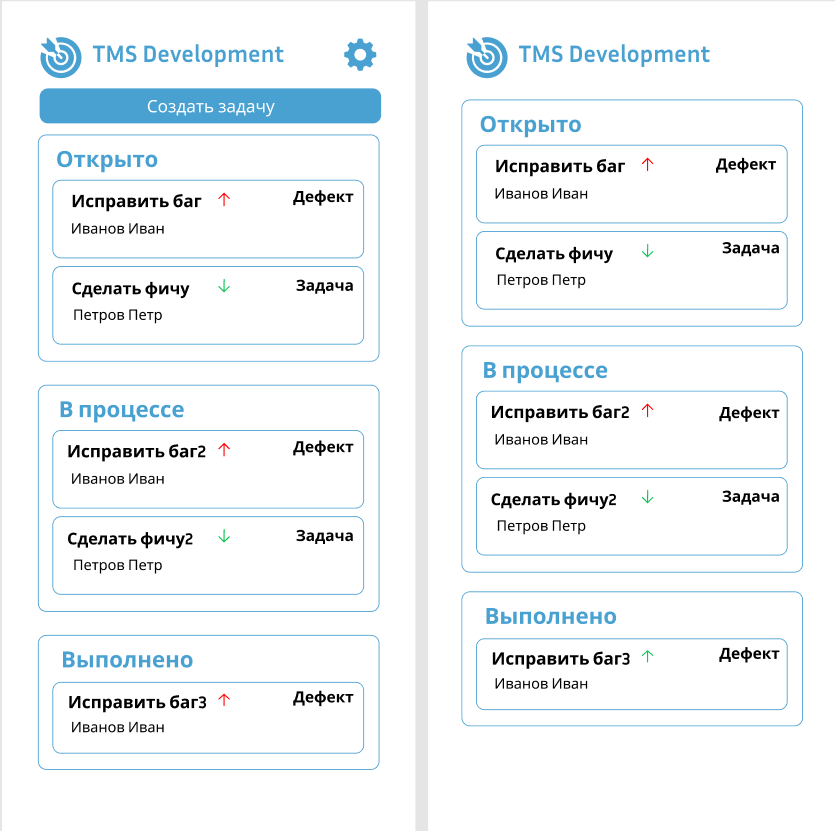
1. подсистема аутентификации, которая отвечает за аутентификацию пользователей;
2. подсистема взаимодействия с БД, которая отвечает за взаимодействие системы с БД;
3. подсистема взаимодействия с серверной частью, которая отвечает за взаимодействие серверной части с клиентской;
4. подсистема «Проект», в состав которой входят следующие подсистемы:
   * подсистема конфигурации проекта, которая отвечает за создание и изменение конфигурации проекта;
   * подсистема конфигурации задачи, которая отвечает за создание и изменение задачи.
   1. Разработка прототипов экранных форм приложения
      1. Мобильное приложение

В данном разделе представлены прототипы интерфейса для пользователя в режиме участника проекта и пользователя в роли менеджера проекта. Т.к. клиент для участника и менеджера проекта будет представлять собой мобильное приложение, прототипы представлены в соответствующем формате.

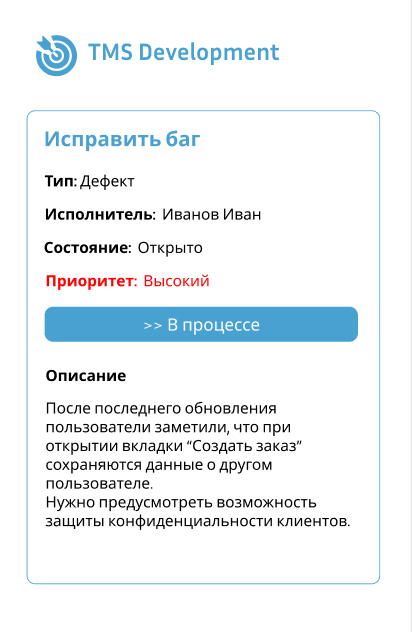
На рисунке 7 представлена форма авторизации для режима участника проекта. Также тут стоить отметить, что данная форма также доступна для менеджеров. На данной форме пользователю предлагается ввести логин и пароль для входа в систему.

  
Рисунок 7 – Прототип формы авторизации для участников и менеджеров проектов

После прохождения авторизации участник проекта переходит на форму списка задач проекта (рисунок 8). Данная форма представляет собой канбан-доску, на которой задачи разделены на категории в зависимости от их состояния. Также на данной доске видны название задач, их исполнитель, тип и приоритет. Для менеджеров на данной форме также доступно создание новой задачи и редактирование проекта.

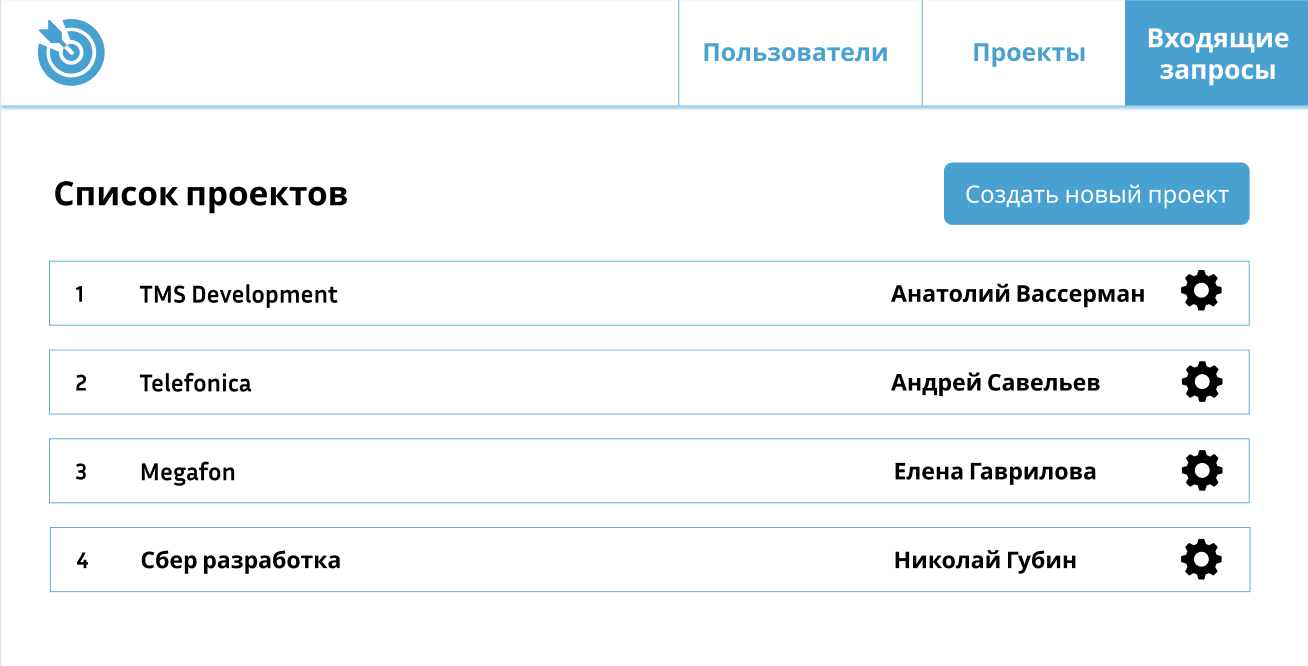
  
Рисунок 8 – Прототип формы списка задач проекта (для менеджера и участника проекта)

При нажатии на определенную задачу пользователь переходит на форму информации о задаче (рисунок 9). На данной форме содержится полная информация о задаче: название, тип, исполнитель, состояние, приоритет и описание. Участнику проекта доступен перевод задачи в следующее состояние. Менеджеру помимо этого доступно редактирование всех характеристик задачи.

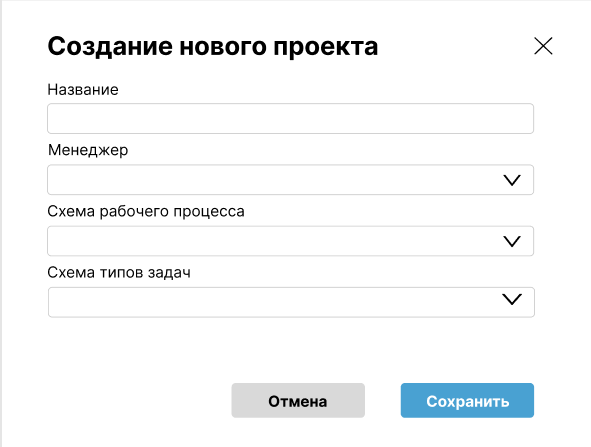
  
Рисунок 9 – Форма информации о задаче

* + 1. Веб-приложение

Режим системного администратора будет обеспечиваться веб-приложением. Для примера работы приложения в режиме системного администратора на рисунке 10 представлен прототип формы списка проектов. Системному администратору доступно создание нового проекта, а также редактирование конкретного проекта.

  
Рисунок 10 – Прототип формы списка проектов

На рисунке 11 представлен прототип формы создания нового проекта. Для создания нового проекта необходимо указать его название, выбрать менеджера из списка пользователей, схему рабочего процесса и схему типов задач.

  
Рисунок 11 – Прототип формы создания проекта

* 1. Разработка информационно-логического проекта системы
     1. Язык UML

Унифицированный язык моделирования (Unified Modeling Language – UML) – это стандартный инструмент для разработки «чертежей» программного обеспечения. Его можно использовать для визуализации, спецификации, конструирования и документирования артефактов программных систем. UML подходит для моделирования любых систем – от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени [9].

В данной работе будем использовать UML для специфицирования системы и ее документирования. Язык UML предоставляет стандартный способ написания проектной документации на системы, включая концептуальные аспекты, такие как бизнес-процессы и функции системы, а также конкретные аспекты, такие как выражения языков программирования, схемы баз данных и повторно используемые компоненты ПО [9].

* + 1. Диаграмма вариантов использования

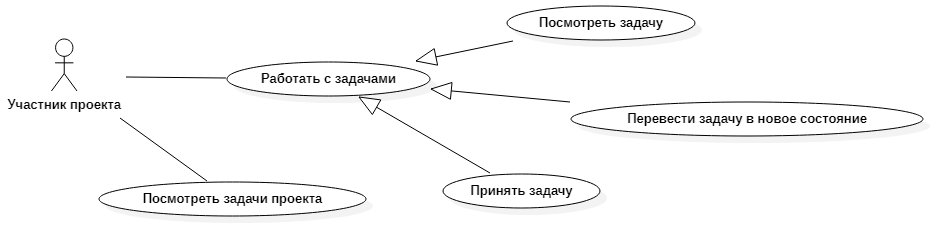
Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актерами и вариантами использования [10].

Актер (actor) – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними.

Вариант использования – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами [10].

Цель спецификации варианта использования заключается в том, чтобы зафиксировать некоторый аспект или фрагмент поведения проектируемой системы без указания особенностей реализации данной функциональности. В этом смысле каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемая система по запросу актера, т. е. определяет один из способов применения системы. Сервис, который инициализируется по запросу актера, должен представлять собой законченную последовательность действий. Это означает, что после того как система закончит обработку запроса актера, она должна возвратиться в исходное состояние, в котором снова готова к выполнению следующих запросов [10].

На рисунке 12 представлена диаграмма вариантов использования для участника проекта. Участнику проекта доступна работа с задачами: просмотр, перевод в новое состояние и подтверждение работы над задачей, а также просмотр списка задач проекта.

  
Рисунок 12 – Диаграмма вариантов использования для участника проекта

На рисунке 13 представлена диаграмма вариантов использования для менеджера проекта. Менеджеру доступна работа с задачами: просмотр, удаление, редактирование и создание; просмотр списка задач проекта и редактирование проекта, а именно задание описания и добавление нового участника проекта.

На рисунке 14 представлена диаграмма вариантов использования для системного администратора. Системному администратору доступны работа с проектами, работа со схемами (схемами типов и схемами рабочего процесса), работа с пользователями, работа с элементами схем (типами и состояниями). Для всех данных действий доступно создание, редактирование и удаление объектов.

* + 1. Диаграмма классов

Диаграммы классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем, они показывают набор классов, интерфейсов и коопераций, а также их связи. На практике диаграммы классов применяют для моделирования статического представления системы, они служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения [11].

На рисунке 15 приведена диаграмма классов системы (этап проектирования). В таблице 1 приведено описание классов.

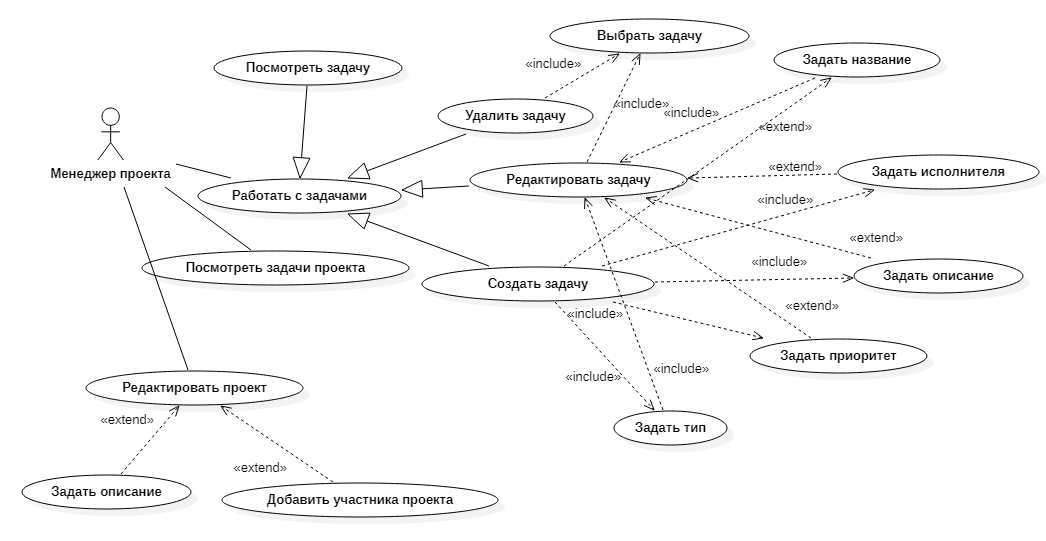
Таблица 1 – Описание классов системы (этап проектирования)

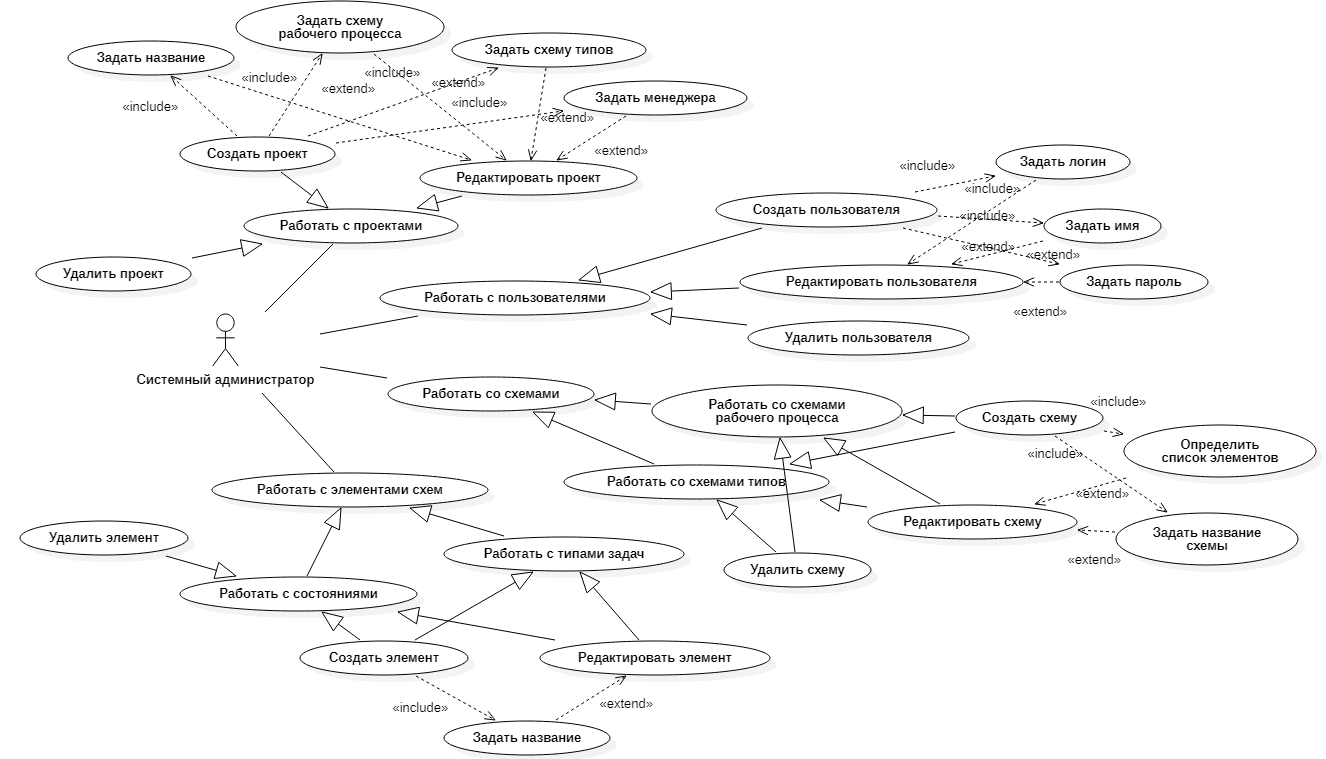
|  |  |
| --- | --- |
| Название класса | Назначение |
| Пользователь | Класс для описания сущности «Пользователь» |
| Проект | Класс для описания сущности «Проект» |
| Задача | Класс для описания сущности «Задача» |
| Тип | Класс для описания сущности «Тип задачи» |
| Схема типов задач | Класс для описания сущности «Схема типов задач» |
| Состояние | Класс для описания сущности «Состояние» |
| Схема рабочего процесса | Класс для описания сущности «Схема рабочего процесса» |
| Репозиторий пользователей | Класс для работы с хранилищем пользователей |
| Репозиторий проектов | Класс для работы с хранилищем проектов |
| Репозиторий задач | Класс для работы с хранилищем задач |
| Репозиторий типов | Класс для работы с хранилищем типов задач |

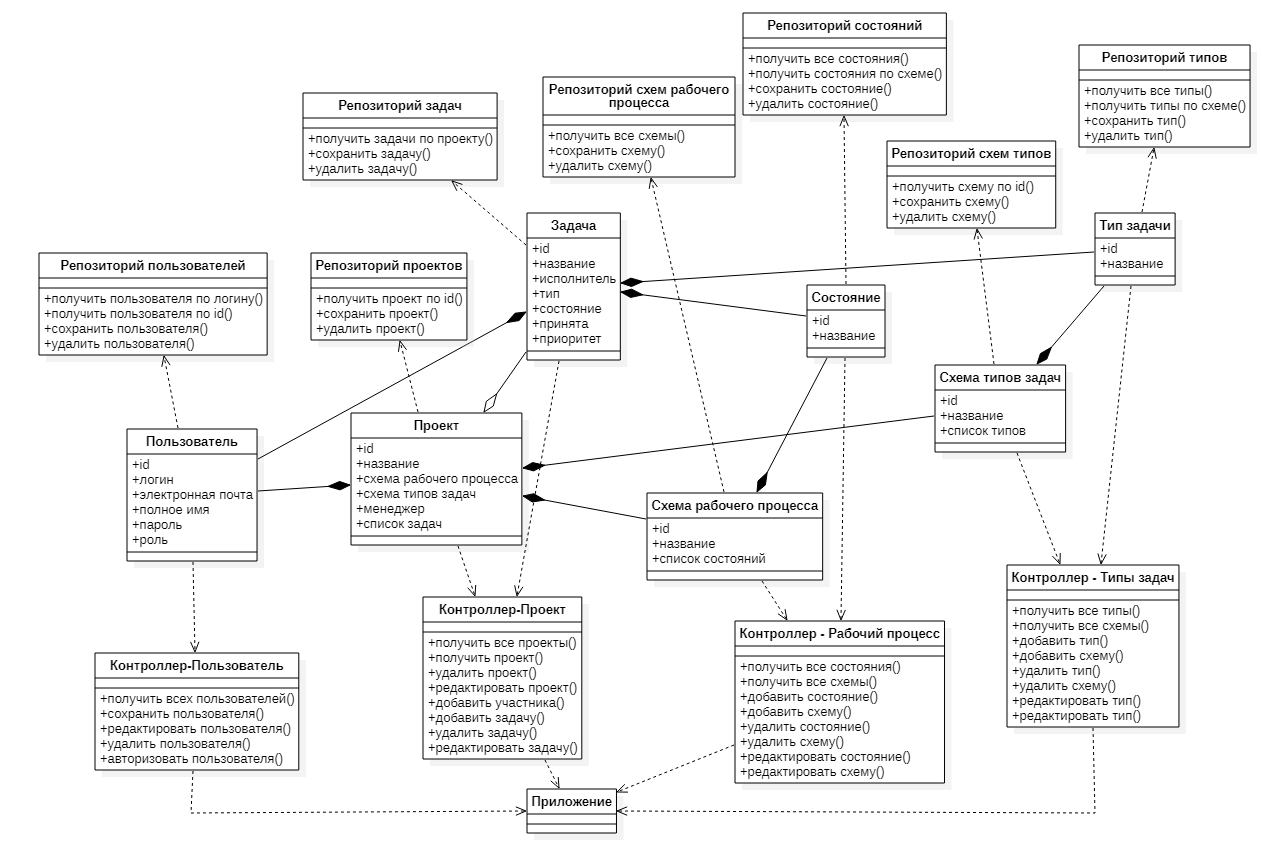
* + 1. Диаграмма деятельности

Диаграмма деятельности – это диаграмма, применяемая в UML для моделирования динамических аспектов систем. По сути, диаграмма деятельности представляет собой блок-схему, которая показывает, как поток управления переходит от одной деятельности к другой.

Моделирование динамических аспектов систем при помощи диаграмм деятельности большей частью подразумевает моделирование последовательных шагов вычислительного процесса. Диаграммы деятельности могут использоваться отдельно для визуализации, специфицирования, конструирования и документирования динамики сообщества объектов либо для моделирования потока управления в операции. Деятельность – это структурированное описание текущего поведения [9].

  
Рисунок 13 – Диаграмма вариантов использования для менеджера проекта

  
Рисунок 14 – Диаграмма вариантов использования для системного администратора

  
Рисунок 15 – Диаграмма классов приложения (этап проектирования)

На рисунке 16 приведена диаграмма деятельности системы в режиме администратора. Работа с системой начинается со страницы авторизации, на которой система считывает логин и пароль, введенные пользователем. В случае успешной авторизации система открывает страницу работы с проектами. Далее, в зависимости от выбора пользователя в меню приложения, система открывает страницу работы с пользователями, рабочим процессом, типами задач или технической поддержки.

На рисунке 17 приведена диаграмм деятельности системы в режиме менеджера и участника проекта. Работа с системой также начинается с формы авторизации. Далее, в зависимости от выбора пользователя в меню приложения, система открывает форму профиля пользователя или форму проектов.

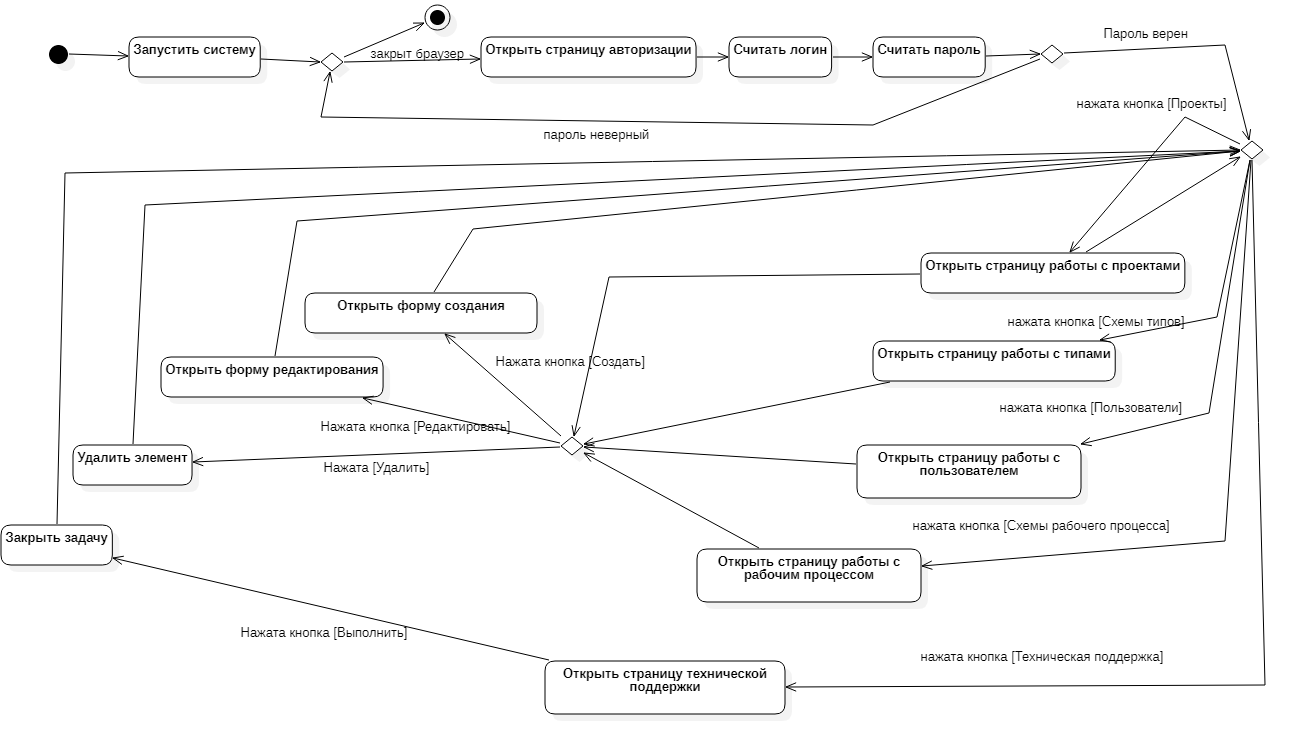
* + 1. Логическая модель данных системы

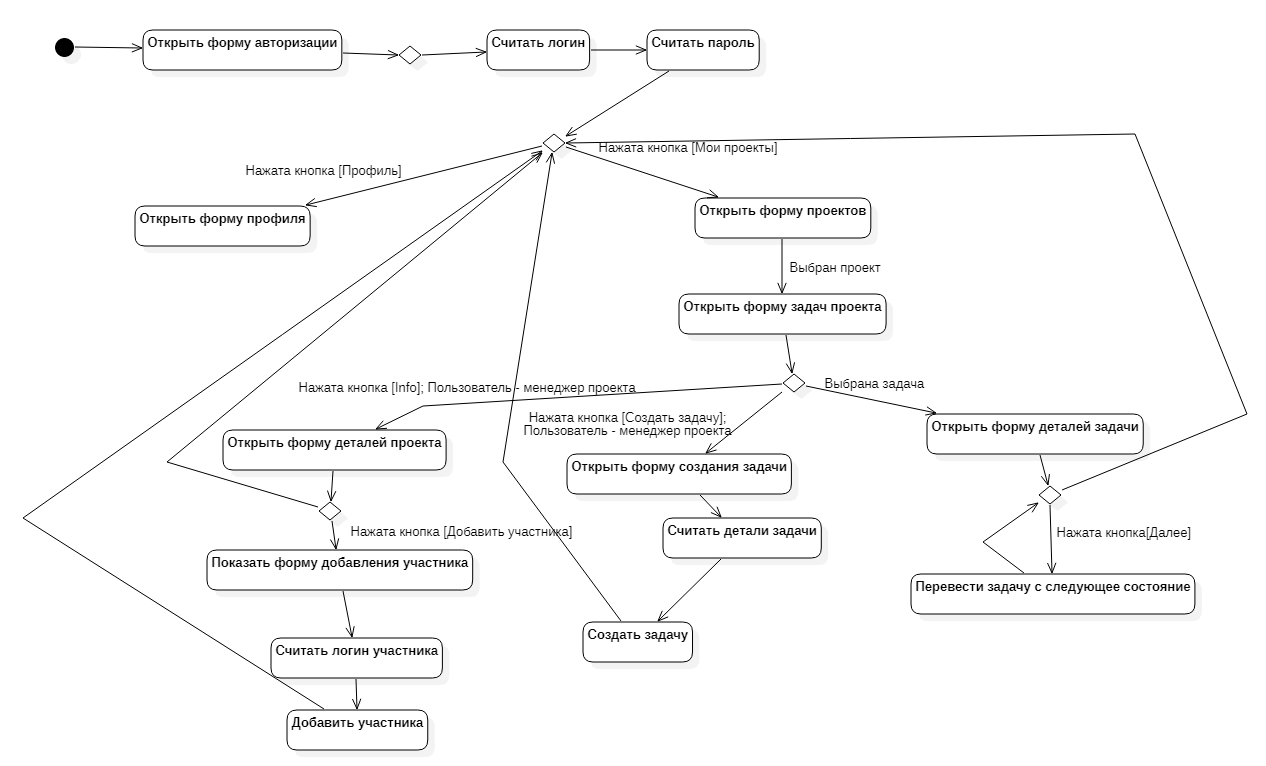
Логическая информационная модель – модель данных, в которой учитывается способ логического хранения данных в памяти ЭВМ. При построении модели базы данных (БД) используются следующие понятия.

Сущность – объект предметной области, который можно отличить от других понятий по некоторым признакам. Сущность состоит из множества своих экземпляров. Каждая сущность обладает свойствами – атрибутами [12].

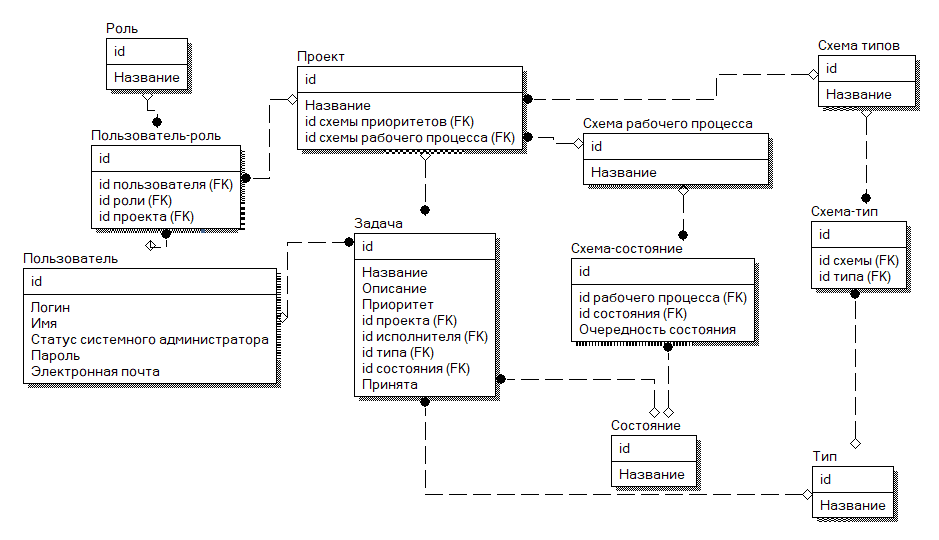
Атрибут – определенное свойство сущности. Именно набор атрибутов, в общем случае уникальный для каждой сущности, позволяет выделить ее среди других объектов и назвать уникальным именем.

Атрибут или набор атрибутов, используемый для идентификации экземпляра сущности, называется ключом сущности. В случае если для идентификации экземпляра используется один атрибут, ключ называется простым; в противном случае ключ составной. Каждый экземпляр сущности однозначно определяется ключом [12].

  
Рисунок 16 – Диаграмма деятельности системы в режиме системного администратора

  
Рисунок 17 – Диаграмма деятельности системы в режиме менеджера проекта

Логическая модель БД разрабатываемой системы приведена на рисунке 17.

  
Рисунок 18 – Логическая модель данных

Описание объектов рассматриваемой предметной области, которые хранятся в базе данных, приведено в таблицах 2-12.

Таблица 2 – Сущность «Пользователь»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID пользователя | Целый | Уникальный идентификатор пользователя |
| Имя | Символьный[50] | Полное имя пользователя, отображаемое в интерфейсе |
| Логин | Символьный[10] | Имя, используемое при идентификации пользователя и его взаимодействии с системой |
| Пароль | Символьный[10] | Пароль пользователя, преобразованный в закодированную строку |
| Email | Символьный[50] | Электронная почта, указанная пользователем при регистрации |

Таблица 3 – Сущность «Роль»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID роли | Целый | Уникальный идентификатор роли |
| Название | Символьный[10] | Название роли |

Таблица 4 – Сущность «Пользователь – Роль»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID сущности | Целый | Уникальный идентификатор сущности |
| ID пользователя | Целый | Уникальный идентификатор пользователя |
| ID роли | Целый | Уникальный идентификатор роли |
| ID проекта | Целый | Уникальный идентификатор проекта |

Таблица 5 – Сущность «Проект»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID проекта | Целый | Уникальный идентификатор проекта |
| Название | Символьный[50] | Название проекта |
| ID схемы рабочего процесса | Целый | Идентификатор схемы рабочего процесса, используемой в проекте |
| ID схемы типов задач | Целый | Идентификатор схемы типов задач, используемой в проекте |

Таблица 6 – Сущность «Задача»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID задачи | Целый | Уникальный идентификатор задачи |
| Название | Символьный[100] | Название задачи |
| Описание | Символьный[500] | Описание задачи |

Таблица 6 – Продолжение таблицы

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Приоритет | Символьный[10] | Приоритет задачи |
| Принята | Логический | Показатель того, принята задача или нет |
| ID проекта | Целый | Уникальный идентификатор проекта |
| ID исполнителя | Целый | Уникальный идентификатор пользователя, ответственного за задачу |
| ID типа | Целый | Уникальный идентификатор типа задачи |
| ID состояния | Целый | Уникальный идентификатор состояния задачи |

Таблица 7 – Сущность «Состояние»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID состояния | Целый | Уникальный идентификатор состояния |
| Название | Символьный[50] | Название состояния |

Таблица 8 – Сущность «Схема рабочего процесса»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID схемы рабочего процесса | Целый | Уникальный идентификатор схемы рабочего процесса |
| Название | Символьный[50] | Название схемы рабочего процесса |

Таблица 9 – Сущность «Схема – Состояние»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID сущности | Целый | Уникальный идентификатор сущности |

Таблица 9 – Продолжение таблицы

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID схемы | Целый | Уникальный идентификатор схемы рабочего процесса |
| ID состояния | Целый | Уникальный идентификатор состояния |
| Очередность | Целый | Очередность состояния в рабочем процессе задачи |

Таблица 10 – Сущность «Тип»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID типа | Целый | Уникальный идентификатор типа задачи |
| Название | Символьный[50] | Название типа задачи |

Таблица 11 – Сущность «Схема типов задач»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID схемы типов | Целый | Уникальный идентификатор схемы типов задач |
| Название | Символьный[50] | Название схемы типов задач |

Таблица 12 – Сущность «Схема – Тип»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID сущности | Целый | Уникальный идентификатор сущности |
| ID схемы | Целый | Уникальный идентификатор схемы типов задач |
| ID типа | Целый | Уникальный идентификатор типа задачи |

* 1. Выбор и обоснование комплекса программных средств

Программное средство – это объект, состоящий из программ, процедур, правил, а также сопутствующих им документации и данных, относящихся к функционированию системы обработки информации [].

Программное средство представляет собой конкретную информацию, объективно существующую как совокупность всех значимых с точки зрения ее представления свойств каждого из материальных объектов, содержащих в фиксированном виде эту информацию [13].

* + 1. Выбор языка программирования

Язык программирования – формальный язык, предназначенный для представления программ [14]. В качестве языка программирования для серверной части приложения был выбран язык программирования Java. Java – язык программирования общего назначения. Относится к объектно-ориентированным языкам программирования, к языкам с сильной типизацией.

Основными преимуществами языка Java являются:

* простота;
* объектно-ориентированный подход;
* безопасность;
* производительность;
* надежность;
* кроссплатформенность.

В качестве языка для разработки клиентской части приложения был выбран язык JavaScript и его фреймворки React JS и React Native. JavaScript – мультипарадигменный язык программирования. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса [15].

React – JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.React может использоваться для разработки веб и мобильных приложений. Его цель – предоставить высокую скорость, простоту и масштабируемость [16].

React Native (также известный как RN) - это популярная платформа мобильных приложений на основе JavaScript, которая позволяет создавать мобильные приложения с собственным интерфейсом для iOS и Android. Фреймворк позволяет создавать приложения для различных платформ, используя одну и ту же кодовую базу [17].

* + 1. Выбор операционной системы

Операционная система – совокупность системных программ, предназначенная для обеспечения определенного уровня эффективности системы обработки информации за счет автоматизированного управления ее работой и предоставляемого пользователю определенного набора услуг [18]. Разработка будет вестись в операционной среде Windows 10. Windows – это операционная система, разработанная корпорацией Microsoft.

На выбор операционной системы Windows повлиял ряд причин:

* масштабирование;
* безопасность;
* отказоустойчивость.

Версия OC Windows 10 отличается от предыдущих следующими  
параметрами:

* поддержка мобильных устройств;
* оптимизированное быстродействие;
* улучшенная безопасность;
* поддержка виртуальных рабочих столов;
* экономия заряда для портативных устройств.
  + 1. Выбор среды программирования

Среда программирования – это набор инструментов, которые используются для преобразования символов в выполнимые вычисления [19].

Обычно среда программирования включает:

1. редактор – средство для создания и изменения исходных файлов, которые содержат написанную на языке программирования программу;
2. компилятор – транслирует символы из исходного файла в объектный модуль, который содержит команды в машинном коде для конкретного компьютера;
3. компоновщик – собирает объектные файлы отдельных компонентов программы и разрешает внешние ссылки от одного компонента к другому, формируя исполняемый файл;
4. отладчик – это средство, которое дает возможность программисту управлять выполнением программы на уровне отдельных команд для диагностики ошибок [19].

В качестве среды разработки будет использоваться Intellij IDEA. IntelliJ IDEA – интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. Начиная с шестой версии продукта, IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными [свободными](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit [20].

Среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, в ней реализована полная поддержка Java SE, Kotlin, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В редакции Ultimate Edition, реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков [20].

Для разработки была выбрана Ultimate-версия в связи с рядом преимуществ:

* бесплатная подписка для студентов;
* поддержка многих языков программирования;
* поддержка фреймворка Spring;
* поддержка инструментов сборки.
  + 1. Выбор СУБД

В качестве СУБД была выбрана PostgreSQL – cвободная объектнореляционная система управления базами данных. PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Также PostgreSQL предоставляет ряд функциональных возможностей, таких как [21]:

* транзакции;
* вложенные запросы;
* представления;
* ссылочная целостность
* внешние ключи;
* сложные блокировки;
* типы, определяемые пользователем;
* наследственность;
* правила.

1. Реализация системы
   1. Разработка и описание интерфейса пользователя

Пользовательский интерфейс, интерфейс пользователя – одна из разновидностей интерфейсов, который является совокупностью средств и методов взаимодействия пользователя с вычислительными устройствами (в частности, ПК) [22].

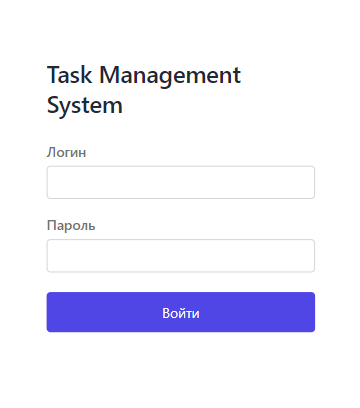
Интерфейс характеризуется удобством, эффективностью, понятностью и дружественностью.

Дружественный интерфейс предоставляет пользователю наиболее удобный способ взаимодействия с программным обеспечением путем обеспечения логичности и простоты в расположении элементов управления [22].

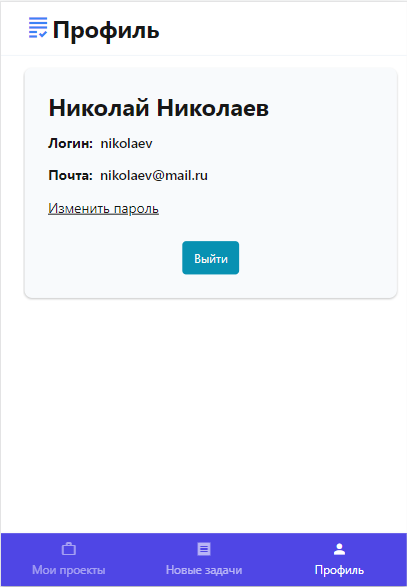
К принципам дружественного интерфейса относят [22]:

* обеспечение права пользователя на ошибку, которое защищают информационно-вычислительные ресурсы системы от непрофессиональных действий на ПК;
* предоставление широкого набора иерархических меню, систему подсказок и обучения и т.п., которые облегчают процесс взаимодействия пользователя с ПК;
* существование системы «отката», которая позволяет при выполнении действия, результаты которого не удовлетворили пользователя, вернуться к предыдущему состоянию системы.
  + 1. Экранные формы мобильного приложения

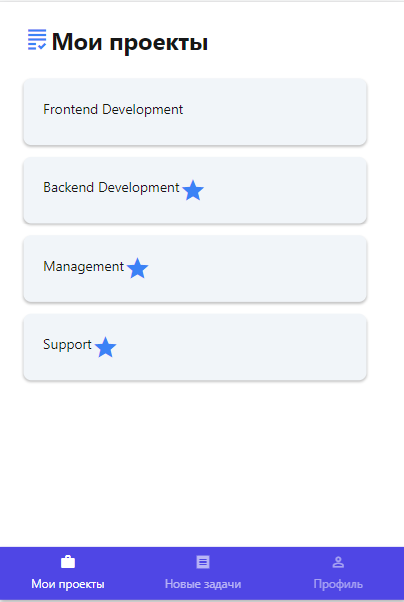
На рисунке 18 представлена экранная форма авторизации для участников и менеджеров проектов. На данной форме пользователю предлагается ввести логин и пароль для прохождения авторизации. В случае неправильных логина или пароль система выводит сообщение об ошибке.

  
Рисунок 18 – Экранная форма авторизации для мобильного приложения

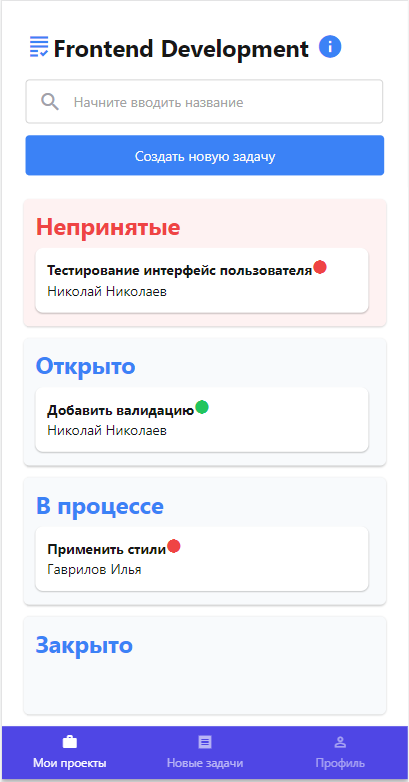
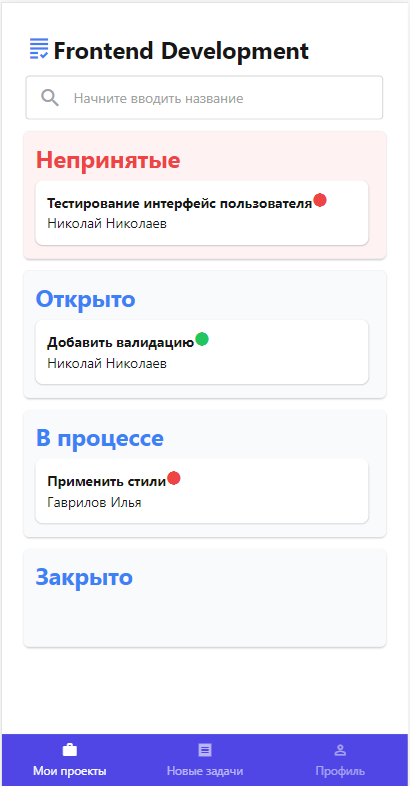
После прохождения авторизации пользователь попадает на страницу личного профиля, которая представлена на рисунке 18. На данной форме представлена информация о пользователе: имя, логин и электронная почта. Также есть возможность сменить пароль и выйти из системы.

  
Рисунок 19 – Экранная форма страницы пользователя

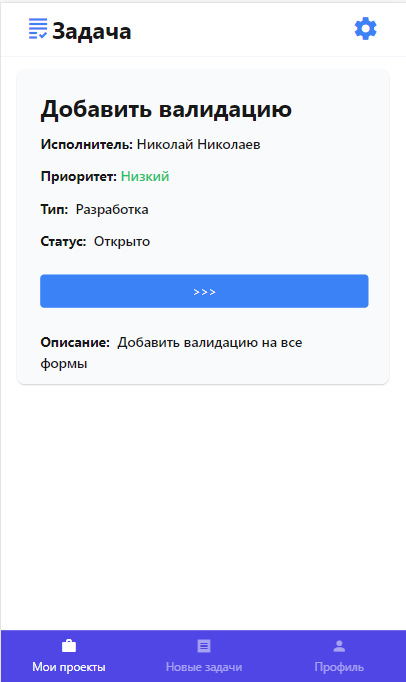
Пользователю также доступна вкладка «Мои проекты», в которой можно посмотреть список проектов, в которых он указан как участник или как менеджер проекта (такие проекты помечаются знаком ). Экранная форма проектов пользователя представлена на рисунке 19.

  
Рисунок 20 – Экранная форма страницы проектов пользователя

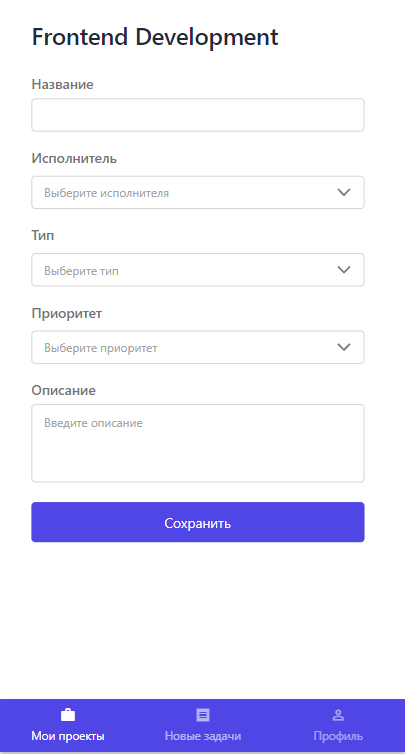
Нажав на определенный проект пользователь попадает на форму вывода всех задач проекта (рисунок 21). На данном экране задачи разделены по состояниям – в соответствие со схемой рабочего процесса. Помимо этого на данной форме присутствует список задач, которые не были просмотрены исполнителем – они находятся в окне «Непринятые». Представленная экранная форма является также точкой доступа к экрану создания новой задачи (рисунок 23) и форме просмотра информации о проекте – по нажатию на кнопку  (рисунок 24). Данные операции доступны только пользователям с правами менеджера проекта.

   
Рисунок 21 – Экранная форма вывода задач проекта

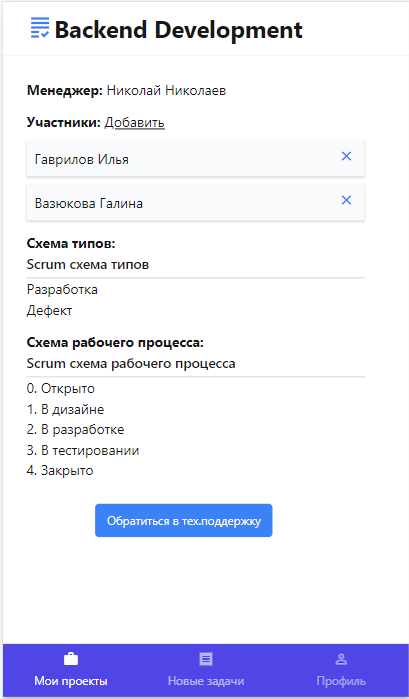
При нажатии на определенную задачу система открывает экран просмотра деталей задачи, который приведен на рисунке 22. На данной форме можно посмотреть все поля задачи (название, исполнителя, приоритет, тип, статус и описание). Исполнителю доступен перевод задачи по состояниям, а также он может принять задачу. После принятия задачи менеджер проекта получает уведомление на свое устройство. Менеджеру также доступно редактирование задачи, экранная форма которого аналогична представленной на рисунке 23.

  
Рисунок 22 – Экран деталей задачи

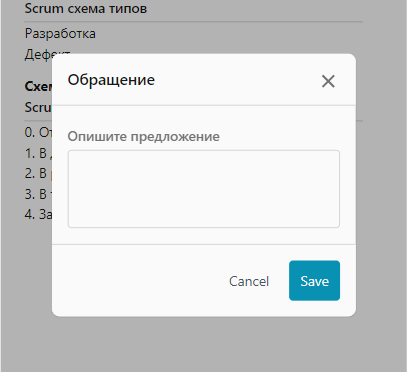
Экран создания задачи приведен на рисунке 23. Для создания задачи необходимо указать ее название, исполнителя (выбирается из списка участников проекта), тип (список типов формируется в соответствие со схемой типов), приоритет и описание. После создания задача попадает в категорию «Непринятые».

  
Рисунок 23 – Экранная форма создания задачи

Менеджеру проекта также доступен просмотр деталей проекта (рисунок 24). На данном экране система выводит информацию о менеджере, участниках проекта, схеме типов и схеме рабочего процесса, которые используются в проекте.

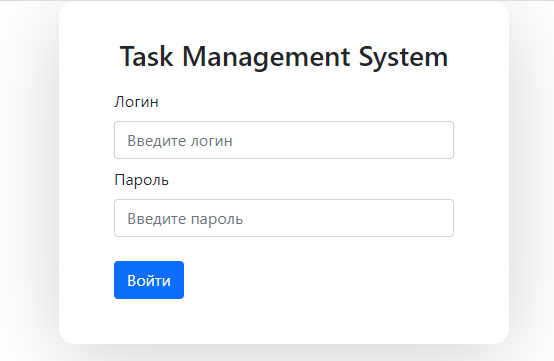
  
Рисунок 24 – Форма вывода информации о проекте

Менеджеру проекта не доступно редактирование проекта (кроме добавления или удаления участников), поэтому в приложении присутствует форма обращения в службу технической поддержки (рисунок 24).

  
Рисунок 25 – Форма обращения в службу технической поддержки

* + 1. Экранные формы веб-приложения

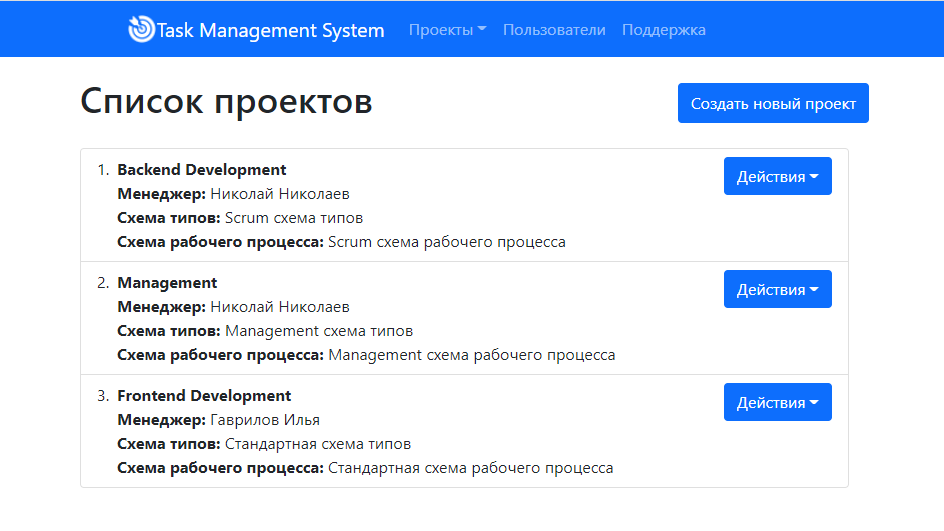
Для работы с приложением пользователю необходимо авторизоваться. Страница авторизации представлена на рисунке 26. Для входа в систему пользователю необходимо ввести логин и пароль.

  
Рисунок 26 – Страница авторизации для веб-приложения

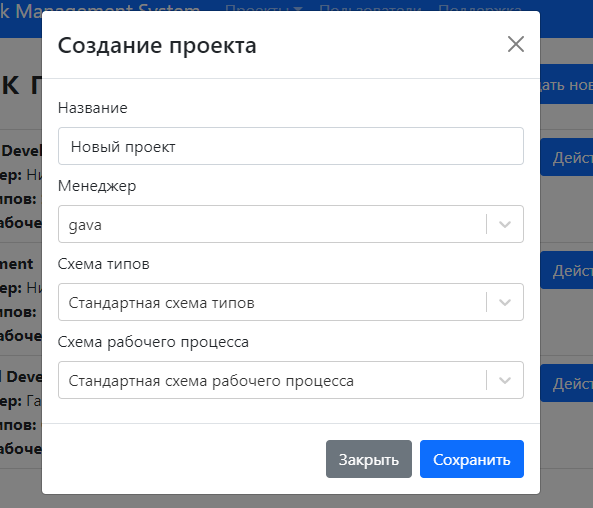
Т.к. доступ к веб-приложению имеют только системные администраторы, пользователи, имеющую иную роль получают сообщение о закрытом доступе (рисунок 27).

  
Рисунок 27 – Сообщение о закрытом доступе

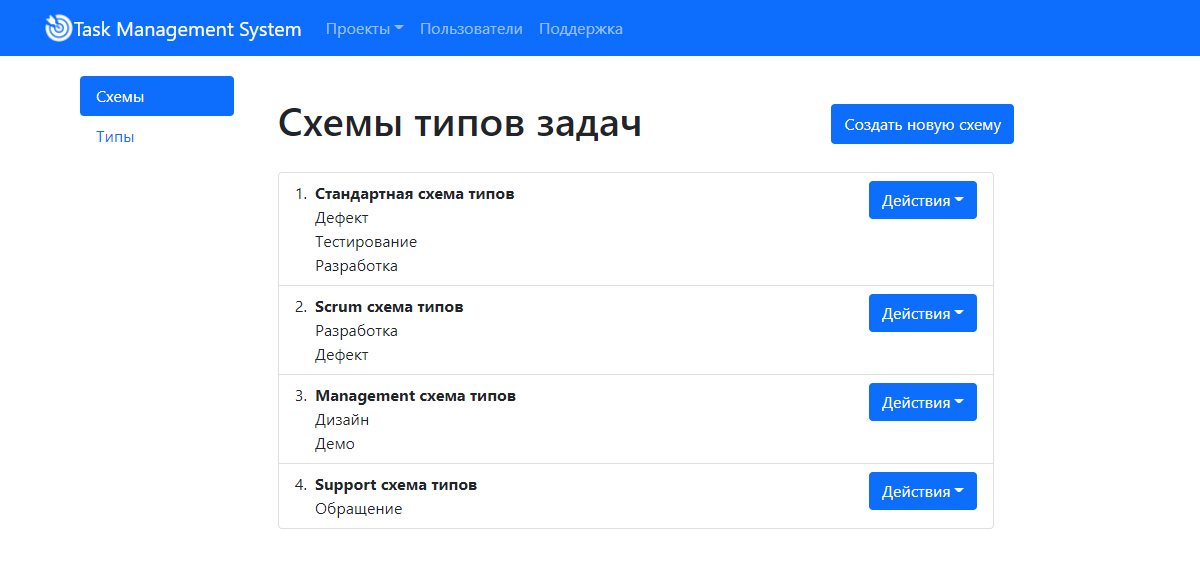
В случае успешной авторизации пользователь попадает на страницу вывода проектов (рисунок 28). На данной странице пользователю доступны следующие функции: просмотр списка проектов, создание нового проекта, редактирование и удаление одного из существующих проектов.

  
Рисунок 28 – Страница работы с проектами

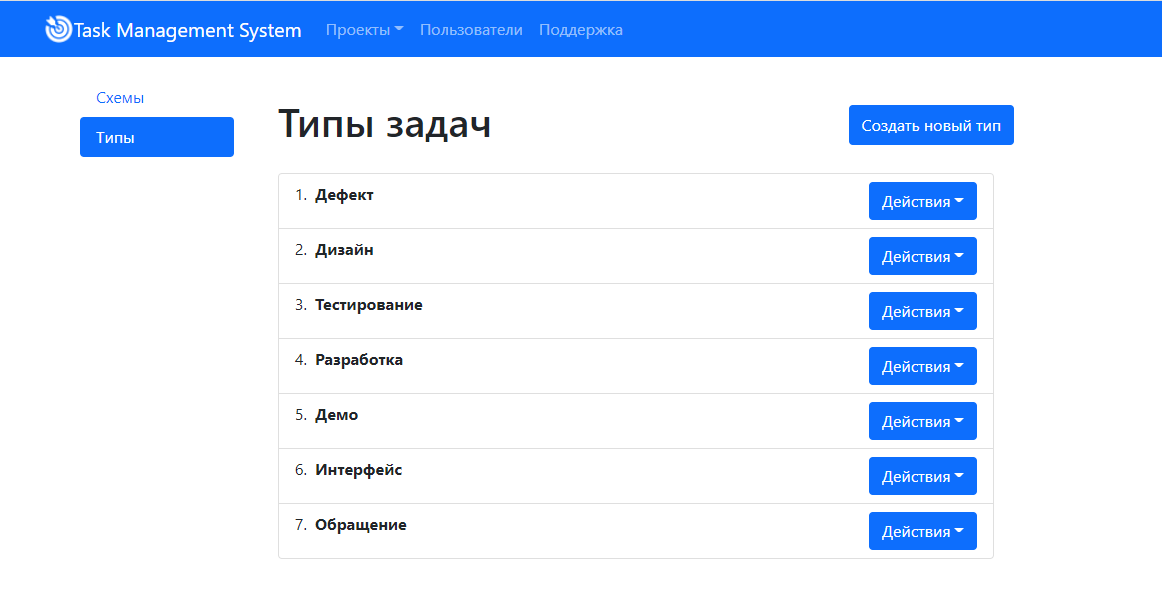
Для создания нового проекта система открывает всплывающее окно. Для создания проекта системному администратору необходимо ввести следующие данные: название, менеджера, схему типов задач и схему рабочего процесса. Форма создания проекта представлена на рисунке 29.

  
Рисунок 29 – Форма создания нового проекта

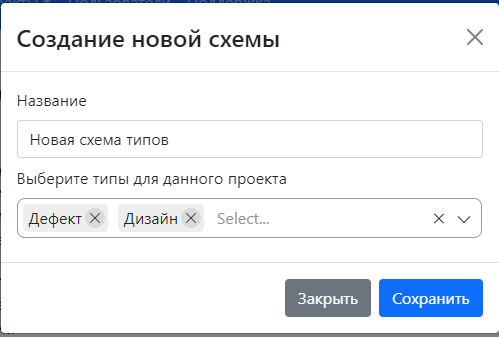
Помимо работы с проектами, системному администратору доступна работа со схемами. На рисунке 30 представлена страница работы со схемами типов задач. Для каждой схемы выводится ее название и набор типов, которые присутствуют в рамках данной схемы. Пользователю также доступно создание, редактирование и удаление схем.

  
Рисунок 30 – Страница работы со схемами типов задач

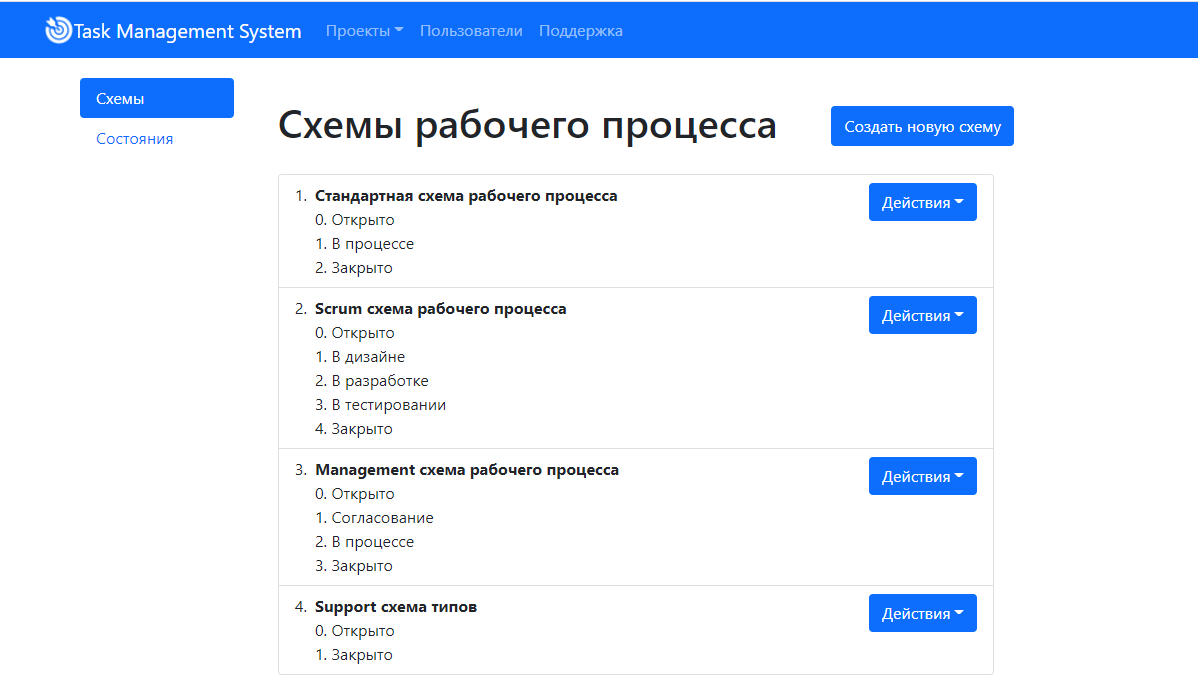
Для создания новой схемы необходимо создать набор типов, которые войдут в данную схему. Страница работы с типами задач представлена на рисунке 31. Пользователю доступны создание, редактирование и удаление типов задач.

  
Рисунок 31 – Страница работы с типами задач

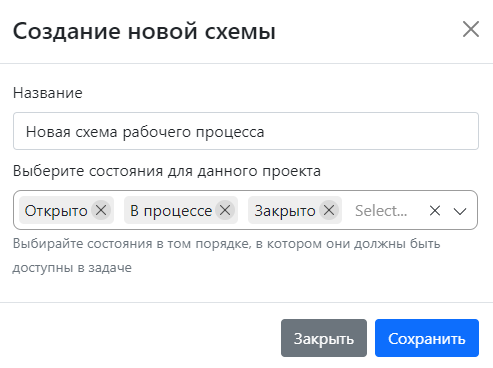
На рисунке 32 представлена форма создания схемы типов задач. Для создания задачи необходимо задать название и список типов задач из списка уже созданных.

  
Рисунок 32 – Форма создания схемы типов задач

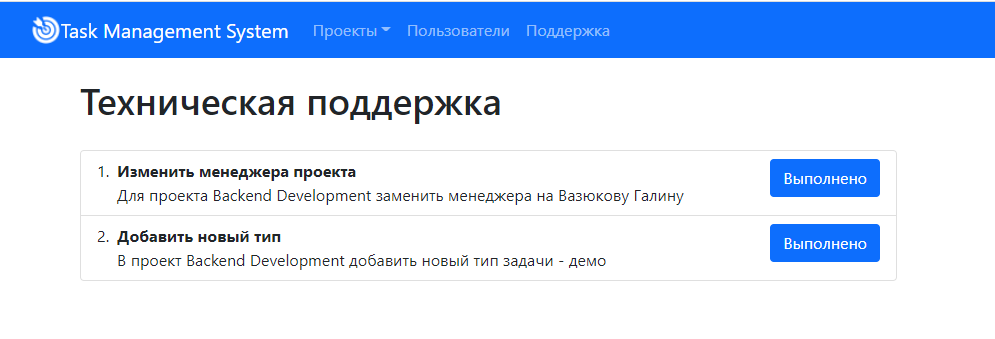
На рисунке 33 представлена страница работы со схемами рабочего процесса. Системному администратору доступны создание, редактирование и удаление схем.

  
Рисунок 33 – Страница работы со схемами рабочего процесса

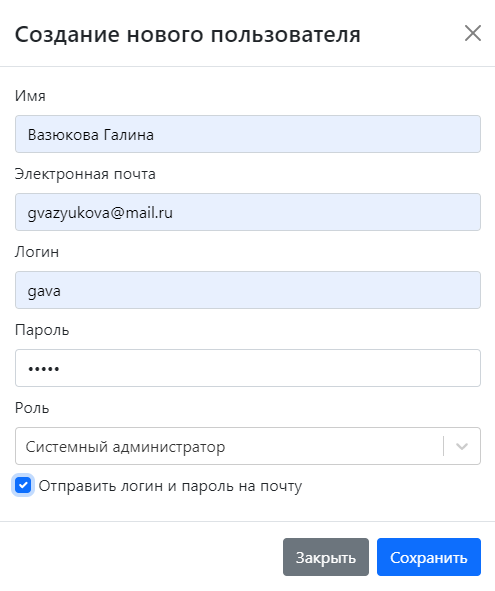
Для создания новой схемы система открывает модальное окно (рисунок 34). Для создания схемы рабочего процесса необходимо задать ее название и список состояний в определенном порядке – в том, в котором они будут доступны при перемещении задачи по состояниям.

  
Рисунок 34 – Форма создания новой схемы рабочего процесса

Для взаимодействия пользователей с системными администраторами реализована служба технической поддержки (рисунок 35). На странице технической поддержки выводятся только новые задачи. После нажатия кнопки «Выполнено» она пропадает из списка.

  
Рисунок 35 – Страница технической поддержки

Системному администратору также доступна работа с пользователями. Т.к. в системе не предусмотрено регистрации, доступ к приложению осуществляется силами системного администратора. На рисунке 36 представлена форма создания нового пользователя. Для выдачи прав доступа к приложению пользователю, необходимо указать его логин, пароль, имя, электронную почту, уровень доступа (системный администратор или пользователь). Есть возможность отправить данные для доступа к приложению пользователю на электронную почту.

  
Рисунок 36 – Форма создания нового пользователя

* 1. Диаграммы реализации

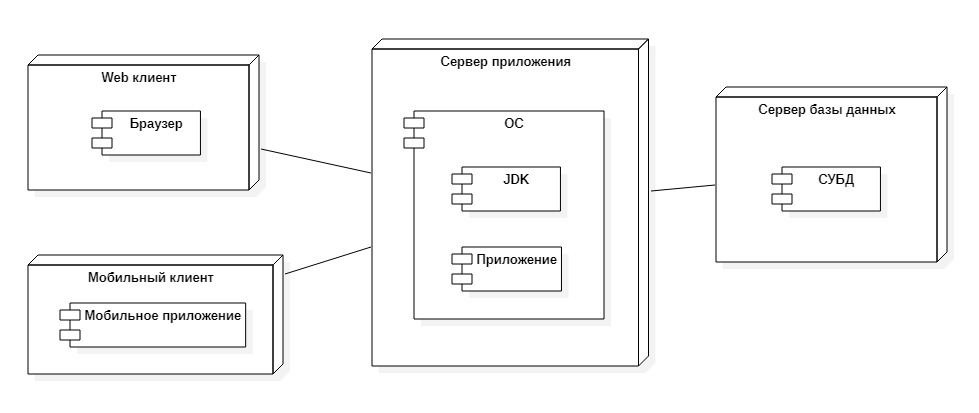
Диаграммы реализации предназначены для отображения состава компилируемых и выполняемых модулей системы, а также связей между ними. Диаграммы реализации разделяются на два конкретных вида: диаграммы компонентов (component diagrams) и диаграммы развертывания (deployment diagrams) [23].

* + 1. Диаграмма развертывания

Диаграмма развертывания – диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

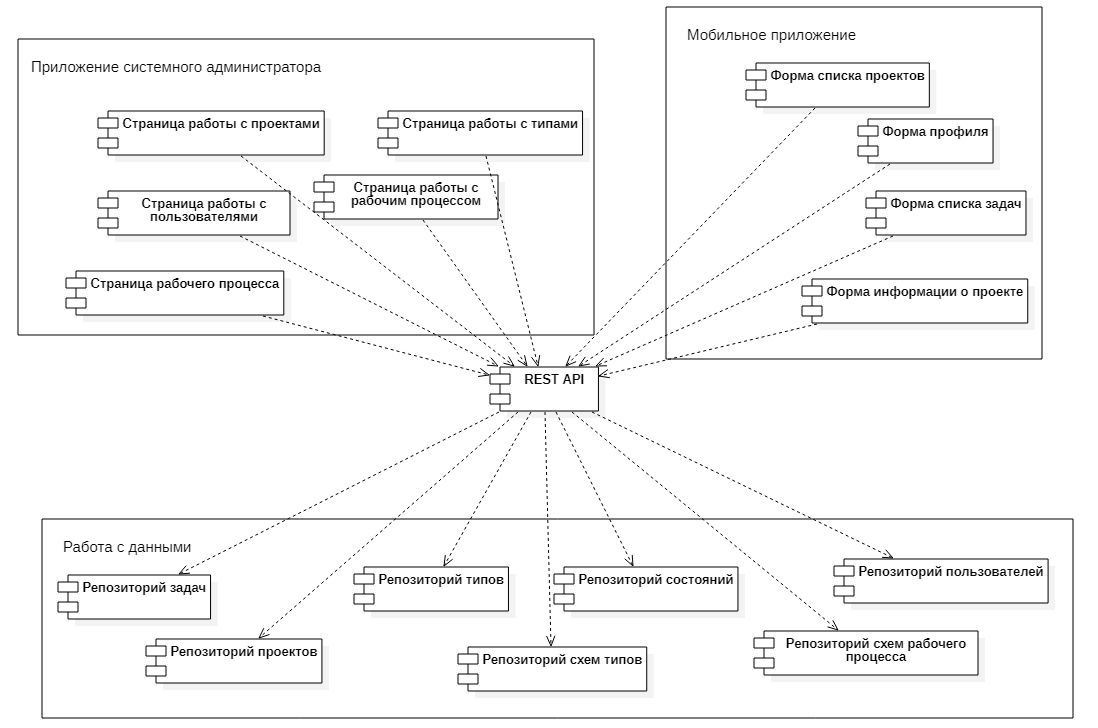
Узел представляет собой физически существующий элемент системы, который может обладать вычислительным ресурсом или являться техническим ресурсом [24].

На рисунке 37 представлена диаграмма развертывания системы.

  
Рисунок 37 – Диаграмма развертывания системы

* + 1. Диаграмма компонентов

На рисунке 38 представлена диаграмма компонентов системы.

  
Рисунок 38 – Диаграмма компонентов системы

* + 1. Диаграмма классов

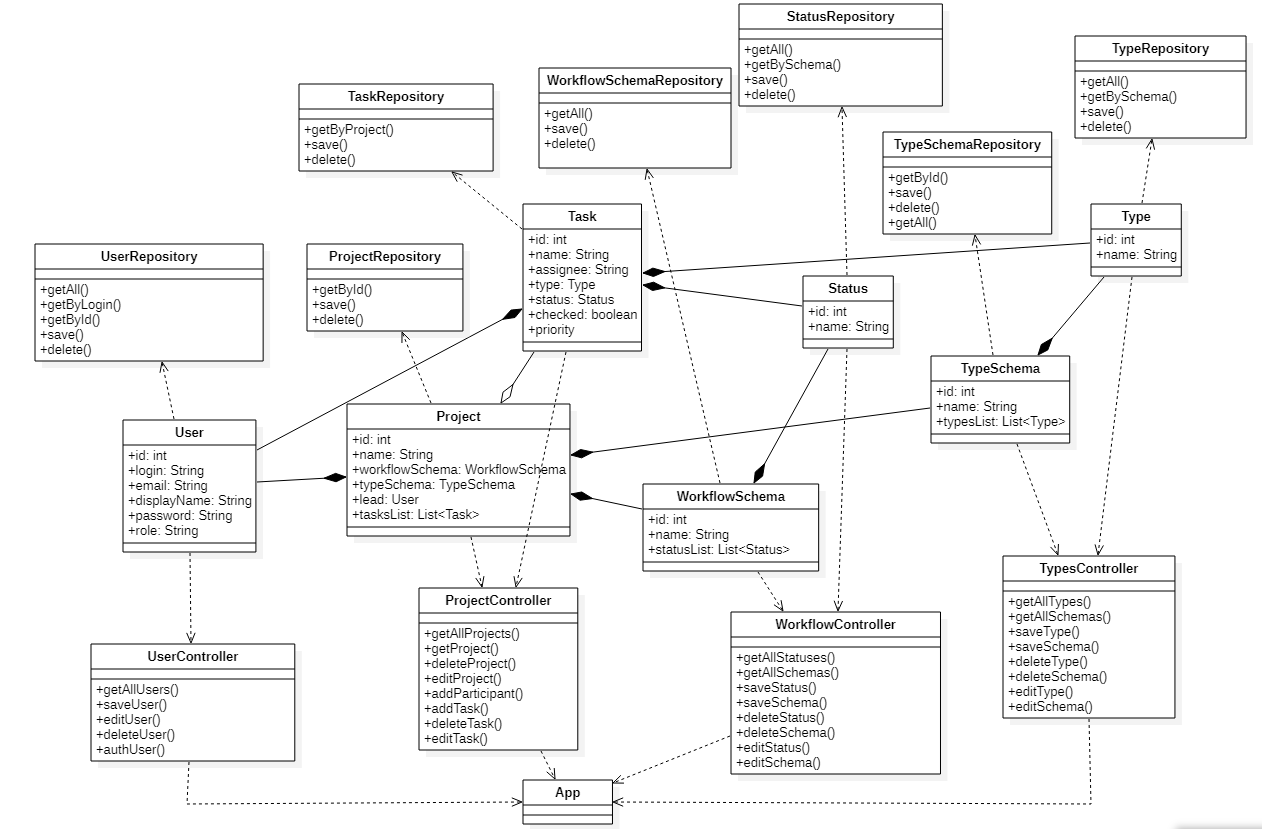
На рисунке 39 представлена диаграмма классов на этапе реализации. Классы приложения можно разделить на три логические части, которые реализуют паттерн MVC – модель, вид, контроллер

* 1. Выбор и обоснование комплекса технических средств
     1. Расчет объема занимаемой памяти. Клиентская часть, использующая мобильное приложение

Расчет объема внешней памяти

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования клиентской части системы, воспользуемся следующей формулой:

VЖД = VОС + VПР,

  
Рисунок 39 – Диаграмма классов системы (этап реализации)

где VОС – объем памяти, занимаемый операционной системой (операционная система Android 11, VОС = 13 Гб);

Vпр – объем памяти, занимаемой файлами программы (Vпр = 70 Мб)

Таким образом, суммарный объем внешней памяти составит:

VЖД = 13 Гб + 70Мб = 13.1 Гб.

Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + Vпр,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

VПР – ОЗУ, которое займет само приложение (не превысит 250 Мб);

Суммарные объемы ОЗУ составит:

VОЗУ = 2 Гб + 250 Мб ~ 2.2 Гб.

Таким образом, 3 Гб оперативной памяти можно счесть минимально необходимым для функционирования клиентской части системы.

Минимальные требования, предъявляемые к системе

Для корректного функционирования клиентской части системы, использующей мобильное приложение, необходимо:

* тип ЭВМ: ARM совместимый;
* объем ОЗУ – не менее 3 Гб;
* процессор – с частотой не менее 1,5 ГГц;
* дисплей с разрешением не менее 800 × 600 пикселей;
* операционная система Android 5.0 или выше (iOS 5.0 или выше).
  + 1. Расчет объема занимаемой памяти. Клиентская часть, использующая веб-приложение

Расчет объема внешней памяти

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования клиентской части системы, воспользуемся следующей формулой:

VЖД = VОС + VСПО + VПР + Vсправки,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 10 Professional 64 бит с пакетом обновлений SP1, VОС = 20 Гб);

VСПО – объем памяти, занимаемый сопутствующим программным обеспечением (JDK, Яндекс Браузер 21.11.3.927; дадим оценку сверху VСПО в 3 Гб);

Vпр – объем памяти, занимаемой файлами программы (Vпр = 150 Мб).

Таким образом, суммарный объем внешней памяти составит:

VЖД = 20 Гб + 3 Гб + 150Мб ~ 23,2 Гб.

Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + Vбраузера + Vпр,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

Vбраузера – ОЗУ, занимаемое браузером (оценим его сверху значением в 1 Гб);

VПР – ОЗУ, которое займет само приложение (не превысит 250 Мб);

Суммарные объемы ОЗУ составит:

VОЗУ = 2 Гб + 1 Гб + 250 Мб ~ 3.2 Гб.

Таким образом, 4 Гб оперативной памяти можно счесть минимально необходимым для функционирования клиентской части системы.

Минимальные требования, предъявляемые к системе

Для корректного функционирования клиентской части системы, использующей веб-приложение, необходимо:

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 4 Гб;
* объем свободного дискового пространства – не менее 30 Гб;
* клавиатура или иное устройство ввода;
* мышь или иное манипулирующее устройство;
* процессор – Intel Core i3 не менее 1,5 ГГц;
* дисплей с разрешением не менее 1024 × 768 пикселей;
* операционная система Windows 7 и выше;
* браузер – Google Chrome 86.0.4240.183 (64-битный) и выше, Яндекс Браузер 21.2.1.108 (64-битный) и выше.
  + 1. Расчет объема занимаемой памяти. Серверная часть

Расчет объема внешней памяти

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования системы, воспользуемся следующей формулой:

VЖД = VОС + VСУБД + VБД,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 10 Professional 64 бит с пакетом обновлений SP1, VОС = 20 Гб);

VСУБД – объем памяти, занимаемый сопутствующим программным обеспечением (для PostgreSQL 13 VСУБД = 2 Гб);

VБД – объем памяти, занимаемый базой данных. Объём памяти, который может быть занят данными, хранящимися в БД неограничен. (минимальный объём, который следует выделить под данные, хранящиеся в БД для корректной работы сервиса возьмём 512 Мб. Возьмем оценку сверху в 2 Гб (VБД = 2 Гб).

Таким образом, суммарный объем внешней памяти составит:

VЖД = 20 Гб + 3 Гб + 2 Гб ~ 25 Гб.

Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + VБД,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

VБД – объем данных из базы, который может быть одновременно загружен в оперативную память (дадим ему оценку сверху в 100 Мб).

Суммарные объемы ОЗУ составит:

VОЗУ = 2 Гб + 100 Мб ~ 2.1 Гб.

Таким образом, 3 Гб оперативной памяти можно счесть минимально необходимым для функционирования серверной части системы.

Минимальные требования, предъявляемые к системе

Для корректного функционирования серверной части системы необходимо:

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 3 Гб;
* объем свободного дискового пространства – не менее 50 Гб;
* клавиатура или иное устройство ввода;
* мышь или иное манипулирующее устройство;
* процессор – Intel Core i3 не менее 1,5 ГГц;
* дисплей с разрешением не менее 1024 × 768 пикселей;
* операционная система Windows 7 и выше;
* PostgreSQL 11 и выше.

Заключение

В процессе выполнения выпускной работы была разработана информационная система управления ИТ-проектами, позволяющая настроить процесс командной работы по методологии Agile.

В первом разделе были приведены основные понятия и определения предметной области, приведены сравнительные характеристики систем-аналогов, на основании этого была сформулирована постановка задачи и основные требования к системе.

Во втором разделе была выбрана архитектура системы, разработана структура системы, спроектированы прототипы интерфейса пользователя, разработан информационно-логический проект системы по методологии UML, в состав которого вошли все канонические диаграммы, логическая модель данных, а также был выбран комплекс программных средств.

В третьем разделе был разработан и описан конечный интерфейс пользователя, выбран и обоснован комплекс технических средств, рассчитан объем занимаемой памяти, необходимый для работы системы, а также были установлены минимальные требования, предъявляемые к системе.

Список использованных источников

1 Плюсы и минусы командной работы [Электронный ресурс]. URL: https://4brain.ru/blog/%D0%BF%D0%BB%D1%8E%D1%81%D1%8B-%D0%B8-%D0%BC%D0%B8%D0%BD%D1%83%D1%81%D1%8B-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B/ (дата обращения 24.12.2021).

2 Методологии управления проектами [Электронный ресурс]. URL: https://www.wrike.com/ru/project-management-guide/metodologii-upravlenija-proektami/ (дата обращения: 24.12.2021).

3 Agile или Waterfall – какой вариант соответствует вашему бизнесу [Электронный ресурс]. URL: https://worksection.com/blog/waterfall-vs-agile.html (дата обращения: 24.12.2021).

4 Что такое Scrum: достоинства и недостатки [Электронный ресурс]. URL: https://sendpulse.by/support/glossary/scrum (дата обращения: 24.12.2021).

5 Что такое Канбан и как внедрить методологии в компании [Электронный ресурс]. URL: https://blog.calltouch.ru/chto-takoe-kanban-i-kak-vnedrit-metodologiyu-v-kompanii/ (дата обращения: 24.12.2021).

6 Jira Sofware [Электронный ресурс]. URL: https://www.atlassian.com/ru/software/jira (дата обращения: 24.12.2021).

7 Что такое Trello и как им пользоваться [Электронный ресурс]. URL: https://netology.ru/blog/trello (дата обращения: 24.12.2021).

8 Архитектура системы [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0\_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B (дата обращения: 25.02.2022).

9 Буч, Г. Язык UML. Руководство пользователя [Текст] /Г. Буч, Д. Рамбо, А. Якобсон. 2-е изд.: Пер. с англ. Мухина Н. М.: ДМК Пресс, 2006. 496 с.: ил.

10 Диаграмма вариантов использования [Электронный ресурс] URL: https://www.intuit.ru/studies/courses/32/32/lecture/1004 (дата обращения: 25.03.2022).

11 Диаграмма классов [Электронный ресурс] // Студопедия: [сайт]. URL: https://studopedia.info/10-59449.html (дата обращения: 25.03.2022).

12 Основные понятия баз данных [Электронный ресурс]. URL: http://inf.susu.ac.ru/Klinachev/lc\_sga\_26.htm (дата обращения: 25.03.2022).

13 ГОСТ 28806-90 Качество программных средств. Термины и определения. М., 1990. 12 с. (Издательство стандартов).

14 ГОСТ 28397-89. Языки программирования. Термины и определения М., 1989. 8 с. (Издательство стандартов).

15 JavaScript: основные понятия [Электронный ресурс]. URL: https://weblecture.ru/node/181 (дата обращения: 25.03.2022).

16 React [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/React (дата обращения: 29.03.2022).

17 Что такое React Native? Комплексное руководство 2021 [Электронный ресурс]. URL: https://habr.com/ru/post/596183/ (дата обращения: 29.03.2022).

18 ГОСТ 15971-90 Системы обработки информации. Термины и определения. М.: Издательство стандартов. 1991. 14 с.

19 Компоненты среды программирования [Электронный ресурс]. – URL: http://www.maksakov-sa.ru/ProgrProd/KompProgrProd/index.html/ (дата обращения: 29.03.2022).

20 IntelliJ IDEA [Электронный ресурс] URL: https://dic.academic.ru/dic.nsf/ruwiki/346106 (дата обращения: 29.03.2022).

Что такое PostgreSQL? Плюсы и минусы бесплатной базы данных [Электронный ресурс] URL: https://oracle-patches.com/common/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-postgresql (дата обращения: 29.03.2022).

22 Пользовательский интерфейс [Электронный ресурс] URL: https://spravochnick.ru/informatika/arhitektura\_personalnogo\_kompyutera/polzovatelskiy\_interfeys/ (дата обращения: 15.11.2018).

23 Диаграммы реализации [Электронный ресурс] URL: http://www.maksakov-sa.ru/ModelUML/DiagrReal/index.html (дата обращения: 15.11.2018).

24 Диаграмма развёртывания [Электронный ресурс] URL: https://www.intuit.ru/studies/courses/32/32/lecture/1024 (дата обращения: 15.11.2018).

ПРИЛОЖЕНИЕ А  
Руководство пользователя

1. Назначение системы

Данная система предназначения для настройки процесса управления проектами по методологии Agile.

1. Условия работы системы

Для корректного функционирования серверной части системы необходимо:

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 3 Гб;
* объем свободного дискового пространства – не менее 50 Гб;
* клавиатура или иное устройство ввода;
* мышь или иное манипулирующее устройство;
* процессор – Intel Core i3 не менее 1,5 ГГц;
* дисплей с разрешением не менее 1024 × 768 пикселей;
* операционная система Windows 7 и выше;
* PostgreSQL 11 и выше.

1. Установка и запуск серверной части системы

Система поставляется в виде jar-архива. Для запуска серверной части необходимо установить СУБД PostgreSQL и контейнер сервлетов Tomcat.

Для запуска системы необходимо открыть jar-файл и запустить сервер приложений с помощью Tomcat.

1. Работа системы
   1. Режим системного администратора

Для доступа к приложению необходимо пройти авторизацию. В случае успешной авторизации у пользователя открывается страница работы с проектами. Пользователям доступно создание нового проекта, а также редактирование и удаление уже существующих. Для создания нового проекта необходимо указать название, менеджера, схему рабочего процесса и схему типов задач.

С помощью меню приложения пользователь может перейти на другие страницы работы с приложением. Первая вкладка – «Пользователи». Данная страница позволяет пользователю создавать, редактировать и удалять пользователей.

Следующая вкладка – «Схемы типов задач». На данной странице пользовалю доступна работа с типами задач и со схемами типов задач соответственно.

Пользователь также может перейти на страницу «Схемы рабочего процесса», с помощью которой можно создавать, редактировать и удалять состояния и схемы рабочего процесса.

По вкладке «Техническая поддержка» системный администратор может посмотреть все открытые запросы пользователей.

* 1. Мобильное приложение

Для доступа к функционалу пользователю необходимо авторизоваться, введя логин и пароль.

В случае успешной авторизации система открывает форму профиля, на которой указаны все данные о текущем пользователе – логин, полное имя и электронная почта.

С помощью меню приложения пользователь может перейти на страницу проектов, в которых он указан в качестве участника или менеджера проекта. Выбрав определенный проект, пользователь переходит на форму задач проекта, на которой система выводит список задач проекта, представленных в виде канбан-доски. Менеджеру доступны также создание задачи и просмотр деталей проекта. На последней форме можно добавить новых участников проекта и написать обращение в техническую поддержку. Участнику проекта данный функционал недоступен.

Выбрав определенную задачу, пользователь может посмотреть детали проекта, а также перевести задачу в следующий статус или принять задачу.

ПРИЛОЖЕНИЕ Б  
Код программы

package com.vkr.tms.controllers;  
  
import com.vkr.tms.model.\*;  
import com.vkr.tms.payload.\*;  
import com.vkr.tms.repositories.\*;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
  
import java.net.URI;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@RestController  
@RequestMapping("/api/projects")  
@CrossOrigin(origins = "\*", maxAge = 3600)  
public class ProjectsController {  
 ProjectRepository projectRepository;  
 TaskRepository taskRepository;  
 PriorityRepository priorityRepository;  
 UserRepository userRepository;  
 RoleMemberRepository roleMemberRepository;  
 TaskTypeRepository taskTypeRepository;  
 TaskTypeSchemaRepository taskTypeSchemaRepository;  
 Type\_SchemaRepository type\_schemaRepository;  
 Status\_SchemaRepository status\_schemaRepository;  
 StatusRepository statusRepository;  
  
 public ProjectsController(ProjectRepository projectRepository, TaskRepository taskRepository, PriorityRepository priorityRepository,  
 UserRepository userRepository, RoleMemberRepository roleMemberRepository, TaskTypeRepository taskTypeRepository,  
 TaskTypeSchemaRepository taskTypeSchemaRepository,  
 Type\_SchemaRepository type\_schemaRepository, Status\_SchemaRepository status\_schemaRepository,  
 StatusRepository statusRepository){  
 this.projectRepository = projectRepository;  
 this.taskRepository = taskRepository;  
 this.priorityRepository = priorityRepository;  
 this.userRepository = userRepository;  
 this.roleMemberRepository = roleMemberRepository;  
 this.taskTypeRepository = taskTypeRepository;  
 this.taskTypeSchemaRepository = taskTypeSchemaRepository;  
 this.type\_schemaRepository = type\_schemaRepository;  
 this.status\_schemaRepository = status\_schemaRepository;  
 this.statusRepository = statusRepository;  
 }  
  
 @GetMapping(value="/all")  
 public ResponseEntity<List<Project>> getAllProjects(){  
 List<Project> projects = projectRepository.findAll();  
 if (projects == null) {  
 return ResponseEntity.*notFound*().build();  
 } else {  
 return new ResponseEntity<>(projects, HttpStatus.*OK*);  
 }  
 }  
  
 @PostMapping(value = "/save", consumes = "application/json", produces = "application/json")  
 public ResponseEntity<Project> saveProject(@RequestBody Project project){  
 Project projectSaved = projectRepository.save(project);  
 if (projectSaved == null){  
 return ResponseEntity.*status*(422).build();  
 }  
 URI uri = ServletUriComponentsBuilder.*fromCurrentRequest*()  
 .buildAndExpand(projectSaved.getId())  
 .toUri();  
 return ResponseEntity.*created*(uri)  
 .body(projectSaved);  
 }  
  
 @GetMapping(value="getByUser/{id}")  
 public ResponseEntity<List<Project>> getProjectsByUser(@PathVariable(name="id") int userId){  
 Usr user = userRepository.getById(userId);  
 List<Project> projects = roleMemberRepository.findByUser(user).stream()  
 .map(RoleMember::getProject)  
 .collect(Collectors.*toList*());  
  
 projects.addAll(projectRepository.findByLead(user));  
 return new ResponseEntity<>(projects, HttpStatus.*OK*);  
  
 }  
  
 @GetMapping(value="projectDetails/{id}")  
 public ResponseEntity<ProjectDetails> getProjectDetails(@PathVariable(name="id") int id){  
 Project project = projectRepository.findById(id).get();  
 List<Usr> participants = roleMemberRepository.findByProject(project).stream()  
 .map(RoleMember::getUser).collect(Collectors.*toList*());  
 System.*out*.println(project.getName());  
 TaskTypeSchema taskTypeSchema = project.getTaskTypeSchema();  
 TaskTypeSchemaBean ttbean = new TaskTypeSchemaBean(taskTypeSchema.getId(), taskTypeSchema.getName(),  
 type\_schemaRepository.getAllByTaskTypeSchema(taskTypeSchema).stream().map(Type\_Schema::getTaskType).collect(Collectors.*toList*()));  
 WorkflowSchema workflowSchema = project.getWorkflowSchema();  
 WorkflowSchemaBean wBean = new WorkflowSchemaBean(workflowSchema.getId(), workflowSchema.getName(),  
 status\_schemaRepository.getAllByWorkflowSchema(workflowSchema).stream()  
 .map(y -> new StatusBean(y.getStatus(), y.getOrdering()))  
 .collect(Collectors.*toList*()));  
 return new ResponseEntity<>(new ProjectDetails(project.getId(), project.getName(), project.getLead(), participants, ttbean, wBean), HttpStatus.*OK*);  
  
 }  
  
 @PostMapping(value="participants/add")  
 public ResponseEntity<RoleMember> saveParticipant(@RequestBody RoleMember participant){  
 RoleMember savedParticipant = roleMemberRepository.save(participant);  
 return new ResponseEntity<>(savedParticipant, HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="participants/delete/{id}")  
 public ResponseEntity<List<RoleMember>> deleteParticipant(@PathVariable(name="id") int id){  
 roleMemberRepository.delete(roleMemberRepository.findById(id).get());  
 return new ResponseEntity<>(roleMemberRepository.findAll(), HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="getUsersNotInProject/{id}")  
 public ResponseEntity<List<Usr>> getUsersNotInProject(@PathVariable(name="id") int id){  
 Project project = projectRepository.findById(id).get();  
 List<Integer> participants = roleMemberRepository.findByProject(project).stream()  
 .map(x -> x.getUser().getId())  
 .collect(Collectors.*toList*());  
 List<Usr> resUsers = userRepository.findAll().stream()  
 .filter(x -> !participants.contains(x.getId()) && project.getLead().getId() != x.getId())  
 .collect(Collectors.*toList*());  
 return new ResponseEntity<>(resUsers, HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="delete/{id}")  
 public ResponseEntity<List<Project>> deleteProject(@PathVariable(name="id") int id){  
 projectRepository.delete(projectRepository.findById(id).get());  
 return getAllProjects();  
 }  
  
 @GetMapping(value="task/{id}")  
 public ResponseEntity<Task> getTask(@PathVariable(name="id") int id){  
 return new ResponseEntity<>(taskRepository.findById(id).get(), HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="{id}/tasks")  
 public ResponseEntity<List<KanbanBean>> getAllTasksByProject(@PathVariable(name="id") int projectId){  
 Project project = projectRepository.findById(projectId).get();  
 List<Task> tasks = new ArrayList<>(taskRepository.getAllByProject(project));  
 List<KanbanBean> beans = new ArrayList<>();  
 beans.add(new KanbanBean("Непринятые", tasks.stream().filter(x -> !x.isChecked()).collect(Collectors.*toList*())));  
 beans.get(0).getTasks().forEach(x -> System.*out*.println(x.getName()));  
 List<Status> statuses = status\_schemaRepository.getAllByWorkflowSchema(project.getWorkflowSchema()).stream().map(Status\_Schema::getStatus).collect(Collectors.*toList*());  
 for (Status status : statuses){  
 beans.add(new KanbanBean(status.getName(), tasks.stream().filter(x -> x.isChecked() && x.getStatus().getId().equals(status.getId())).collect(Collectors.*toList*())));  
 }  
  
 return new ResponseEntity<>(beans, HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="/mytasks/{id}")  
 public ResponseEntity<List<KanbanBean>> getMyTasks(@PathVariable(name="id") int userId){  
 Usr user = userRepository.getById(userId);  
 List<Task> tasks = new ArrayList<>(taskRepository.getAllByAssignee(user));  
 List<KanbanBean> beans = new ArrayList<>();  
 beans.add(new KanbanBean("Непринятые", tasks.stream().filter(x -> !x.isChecked()).collect(Collectors.*toList*())));  
  
 return new ResponseEntity<>(beans, HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="priorities")  
 public ResponseEntity<List<Priority>> getAllPriorities(){  
 return new ResponseEntity<>(priorityRepository.findAll(), HttpStatus.*OK*);  
 }  
  
 @PostMapping(value="saveTask")  
 public ResponseEntity<Task> saveTask(@RequestBody Task task){  
 Task savedTask = taskRepository.save(task);  
 URI uri = ServletUriComponentsBuilder.*fromCurrentRequest*()  
 .buildAndExpand(savedTask.getId())  
 .toUri();  
 return ResponseEntity.*created*(uri)  
 .body(savedTask);  
 }  
  
 @GetMapping(value="checkTask/{id}")  
 public ResponseEntity<Task> checkTask(@PathVariable(name="id") int id){  
 Task task = taskRepository.findById(id).get();  
 task.setChecked(true);  
 Task savedTask = taskRepository.save(task);  
 return new ResponseEntity<>(savedTask, HttpStatus.*OK*);  
 }  
  
 @GetMapping(value="moveTask/{id}")  
 public ResponseEntity<Task> moveTask(@PathVariable(name="id") int id){  
 Task task = taskRepository.findById(id).get();  
 Project project = task.getProject();  
 WorkflowSchema workflowSchema = project.getWorkflowSchema();  
 List<Status\_Schema> status\_schemas = status\_schemaRepository.getAllByWorkflowSchema(workflowSchema);  
 int o = 0;  
 for (Status\_Schema status\_schema : status\_schemas){  
 if (status\_schema.getStatus().getId() == task.getStatus().getId()){  
 o = status\_schema.getOrdering();  
 break;  
 }  
 }  
 final int t = o + 1;  
 Status targetStatus = status\_schemas.stream().filter(x -> x.getOrdering() == t).collect(Collectors.*toList*()).get(0).getStatus();  
 task.setStatus(targetStatus);  
 Task savedTask = taskRepository.save(task);  
 return new ResponseEntity<>(savedTask, HttpStatus.*OK*);  
 }  
  
 @PostMapping(value="saveSupportTask")  
 public ResponseEntity<Task> saveSupportTask(@RequestBody Task task){  
 Project supportProject = projectRepository.findById(1).get();  
 TaskType type = type\_schemaRepository.getAllByTaskTypeSchema(supportProject.getTaskTypeSchema()).get(0).getTaskType();  
 Status status = status\_schemaRepository.getAllByWorkflowSchema(supportProject.getWorkflowSchema()).stream().filter(x -> x.getOrdering() == 0).findFirst().get().getStatus();  
 task.setPriority(priorityRepository.getByName("Высокий"));  
 task.setChecked(true);  
 task.setStatus(status);  
 task.setProject(supportProject);  
 task.setType(type);  
 task.setAssignee(supportProject.getLead());  
 return saveTask(task);  
 }  
}

import React, { useState, useEffect } from 'react'

import { ListGroup, Button, Modal, Form, Dropdown, Alert } from 'react-bootstrap'

import Select from 'react-select';

import '../App.css';

import axios from "axios"

import Header from '../Components/Header';

export default function ProjectsList(){

    const [flag, setFlag] = useState(false);

    const [project, setProject] = useState({

        id: 0,

        name: "",

        lead: {

            id:0,

            login:""

        },

        workflowSchema: 0,

        taskTypeSchema: 0

    });

    const [projects, setProjects] = useState([]);

    const [usersOptions, setUsers] = useState([]);

    const [users, setU] = useState([]);

    const [typesOptions, setTaskTypes] = useState([]);

    const [workflowsOptions, setWorkflows] = useState([]);

    const [show, setShow] = useState(false);

    const [showAlert, setShowAlert] = useState(true);

    const handleClose = () => setShow(false);

    const handleShow = () => setShow(true);

    const openNewProject = () => {

        setProject({

            id: 0,

            name: "",

            lead: {

                id:0,

                login:""

            },

            workflowSchema: 0,

            taskTypeSchema: 0

        })

        console.log(project)

        handleShow();

    }

    const saveProject = () =>{

        handleClose();

        axios.post(`http://localhost:5788/api/projects/save`, project)

            .then((res) => {

                setFlag(!flag)

            }).catch((error) => {

                console.log(error)

            });

        //setUser({})

    }

    const showDeleteAlert = (project) => {

        setShowAlert(true)

    }

    const editProject = (project) => {

        setProject(project);

        setShow(true);

    }

    const deleteProject = id =>{

        axios.get(`http://localhost:5788/api/projects/delete/` + Number(id))

            .then(response => {

                setProjects(response.data);

                setFlag(!flag)

            })

            .catch((error) => {

                console.log(error);

            })

    }

    useEffect(() => {

        let data = {

            headers: {

                "Content-Type": "application/json"

            }

        }

        axios.get(`http://localhost:5788/api/projects/all`, data)

            .then(response => {

                setProjects(response.data);

                //setFlag(false)

            })

            .catch((error) => {

                console.log(error);

            })

    }, [flag])

    useEffect(() => {

        let data = {

            headers: {

                "Content-Type": "application/json"

            }

        }

        axios.get(`http://localhost:5788/api/users/all`, data)

            .then(response => {

                setU(response.data)

                const opt = response.data.map((element) => {

                    return {label: element.username, value: element.id}

                })

                console.log(opt)

                setUsers(opt);

                //setFlag(false)

            })

            .catch((error) => {

                console.log(error);

            })

    }, [show])

    useEffect(() => {

        let data = {

            headers: {

                "Content-Type": "application/json"

            }

        }

        axios.get(`http://localhost:5788/api/tasktype/schemas`, data)

            .then(response => {

                const opt = response.data.map((element) => {

                    return {label: element.name, value: element.id}

                })

                console.log(opt)

                setTaskTypes(opt);

                //setFlag(false)

            })

            .catch((error) => {

                console.log(error);

            })

    }, [show])

    useEffect(() => {

        let data = {

            headers: {

                "Content-Type": "application/json"

            }

        }

        axios.get(`http://localhost:5788/api/workflow/schemes`, data)

            .then(response => {

                const opt = response.data.map((element) => {

                    return {label: element.name, value: element.id}

                })

                console.log(opt)

                setWorkflows(opt);

                //setFlag(false)

            })

            .catch((error) => {

                console.log(error);

            })

    }, [show])

    useEffect(() => {

        let data = {

            headers: {

                "Content-Type": "application/json"

            }

        }

        axios.get(`http://localhost:5788/api/projects/all`, data)

            .then(response => {

                setProjects(response.data);

                //setFlag(false)

            })

            .catch((error) => {

                console.log(error);

            })

    }, [flag])

    const handleName = event => {

        project.name = event.target.value;

    }

    const handleLead = val => {

        let selectedUser = users.find(e => e.id == val.value);

        project.lead = selectedUser;

    }

    const handleTypesSchema = val =>{

        project.taskTypeSchema = {

            id: Number(val.value),

            name: val.label

        };

        console.log(project)

    }

    const handleWorkflowSchema = val =>{

        project.workflowSchema = {

            id: Number(val.value),

            name: val.label

        };

        console.log(project)

    }

        return (

            <>

            <Header/>

            <div className='content-main'>

                <div className="flexClass">

                    <h1>Список проектов</h1>

                    <Button bg="primary" className='leftPosition' onClick={openNewProject}> Создать новый проект</Button>

                </div>

                <ListGroup as="ol" numbered className="content">

                    {projects.map(project => (

                        project.id !== 1 &&

                        <ListGroup.Item

                            as="li"

                            className="d-flex justify-content-between align-items-start"

                        >

                            <div className="ms-2 me-auto">

                            <div className="fw-bold">{project.name}</div>

                            <div><b>Менеджер:</b> {project.lead.displayName}</div>

                            <div><b>Схема типов:</b> {project.taskTypeSchema.name}</div>

                            <div><b>Схема рабочего процесса:</b> {project.workflowSchema.name}</div>

                            </div>

                            <Dropdown>

                                <Dropdown.Toggle id="dropdown-basic">

                                    Действия

                                </Dropdown.Toggle>

                                <Dropdown.Menu>

                                    <Dropdown.Item onClick={() => editProject(project)}>Редактировать</Dropdown.Item>

                                    <Dropdown.Item onClick={() => deleteProject(project.id)}>Удалить</Dropdown.Item>

                                </Dropdown.Menu>

                            </Dropdown>

                        </ListGroup.Item>

                    ))}

                </ListGroup>

                <Modal show={show} onHide={handleClose}>

                    <Modal.Header closeButton>

                        <Modal.Title>{project !== undefined ? 'Редактирование' : 'Создание нового' } проекта</Modal.Title>

                    </Modal.Header>

                    <Modal.Body>

                        <Form>

                            <Form.Group className="mb-2" controlId="">

                                <Form.Label> Название </Form.Label>

                                <Form.Control placeholder='Введите название проекта' onChange={handleName} defaultValue={project !== undefined ? project.name : ""}/>

                            </Form.Group>

                            <Form.Group className="mb-2" controlId="">

                                <Form.Label> Менеджер </Form.Label>

                                <Select

                                    options={usersOptions}

                                    onChange={handleLead}

                                    defaultValue={project !== undefined ? usersOptions.find(e => e.value == project.lead.id) : ""}

                                />

                            </Form.Group>

                            <Form.Group className="mb-2" controlId="">

                                <Form.Label> Схема типов </Form.Label>

                                <Select

                                    options={typesOptions}

                                    onChange={handleTypesSchema}

                                    defaultValue={project !== undefined ? typesOptions.find(e => e.value == project.taskTypeSchema.id) : ""}

                                />

                            </Form.Group>

                            <Form.Group className="mb-2" controlId="">

                                <Form.Label> Схема рабочего процесса </Form.Label>

                                <Select

                                    options={workflowsOptions}

                                    onChange={handleWorkflowSchema}

                                    defaultValue={project !== undefined ? workflowsOptions.find(e => e.value == project.workflowSchema.id) : ""}

                                />

                            </Form.Group>

                        </Form>

                    </Modal.Body>

                    <Modal.Footer>

                    <Button variant="secondary" onClick={handleClose}>

                        Закрыть

                    </Button>

                    <Button variant="primary" onClick={saveProject}>

                        Сохранить

                    </Button>

                    </Modal.Footer>

                </Modal>

            </div>

            </>

        )

    }

import {useEffect, useState} from 'react'

import {StyleSheet} from 'react-native'

import {NativeBaseProvider, Box, Button, Modal, Container, Heading, Text, FormControl,Input, HStack, Icon, Pressable, Center} from 'native-base'

import {MaterialCommunityIcons, MaterialIcons} from 'react-native-vector-icons'

import axios from 'axios';

export default function Task(props) {

    const [loaded, setLoaded] = useState(false);

    const [flag, setFlag] = useState(false);

    const [task, setTask] = useState({});

    let taskId = JSON.parse(localStorage.getItem('task')).id;

    console.log(taskId)

    const BASE\_API\_URL = "http://localhost:5788/api/projects/";

    useEffect(() => {

        axios.get(BASE\_API\_URL + "task/" + taskId)

            .then(response => {

                setTask(response.data);

                setLoaded(true);

            })

            .catch((error) => {

                console.log(error);

            })

        }, [flag])

    const checkTask = () =>{

        if (task.checked == false){

            axios.get(BASE\_API\_URL + "checkTask/" + task.id)

            .then(response => {

                setTask(response.data);

            })

        }

        else{

        axios.get(BASE\_API\_URL + "moveTask/" + task.id)

            .then(response => {

                setTask(response.data);

            })

        }

    }

    return (

        <>

        {loaded &&

        <NativeBaseProvider>

            <Box flex={1} bg="white" safeAreaTop width="100%" alignSelf="center">

                <Container minHeight="640" width="100%">

                    <HStack pt="3" pl="6" pb="3" mb="3" w="414" borderBottomColor="blueGray.100" borderBottomWidth="1" >

                        <Icon as={<MaterialIcons name="grading" />} color="blue.500" size="xl" />

                        <Heading>Задача</Heading>

                        <Icon as={<MaterialIcons name="settings" />} color="blue.500" size="xl" ml="225"/>

                    </HStack>

                    <Box alignItems="center">

                        <Box minW="380" ml="4" rounded="lg" overflow="hidden" borderWidth="1" \_dark={{

                            borderColor: "coolGray.600",

                            backgroundColor: "gray.700"

                            }} \_web={{

                            shadow: 2,

                            borderWidth: 0

                            }} \_light={{

                            backgroundColor: "blueGray.50"

                            }}>

                            <Container mt="6" ml="6">

                                <Heading>{task.name} </Heading>

                                <Text mt="3" fontWeight="medium">

                                    <Text bold>Исполнитель: </Text>

                                    {task.assignee.displayName}

                                </Text>

                                <Text mt="3" fontWeight="medium">

                                    <Text bold>Приоритет: </Text>

                                    <Text color="green.500">{task.priority.name}</Text>

                                </Text>

                                <Text mt="3" fontWeight="medium">

                                    <Text bold>Тип: </Text> {task.type.name}

                                </Text>

                                <Text mt="3" fontWeight="medium">

                                    <Text bold>Статус: </Text> {task.status.name}

                                </Text>

                                <Box alignItems="center" mt="6" mb="6">

                                    <Button size="sm" bg="blue.500" w="335" onPress={checkTask}>{task.checked ? '>>>' : 'Принять' }</Button>

                                </Box>

                                <Text mb="3" fontWeight="medium">

                                    <Text bold>Описание: </Text> {task.description}

                                </Text>

                            </Container>

                        </Box>

                    </Box>

                </Container>

                    <HStack bg="indigo.600" alignItems="center" safeAreaBottom shadow={6}>

                    <Pressable cursor="pointer" opacity="1" py="2" flex={1} onPress={() => props.navigation.navigate('Projects')}>

                        <Center>

                        <Icon mb="1" as={<MaterialIcons name="work" />} color="white" size="sm" />

                        <Text color="white" fontSize="12">

                            Мои проекты

                        </Text>

                        </Center>

                    </Pressable>

                    <Pressable cursor="pointer" opacity="0.5" py="2" flex={1} >

                        <Center>

                        <Icon mb="1" as={<MaterialIcons name={"receipt"} />} color="white" size="sm" onPress={() => props.navigation.navigate('Tasks')}/>

                        <Text color="white" fontSize="12">

                            Новые задачи

                        </Text>

                        </Center>

                    </Pressable>

                    <Pressable cursor="pointer" opacity="0.5" py="2" flex={1} >

                        <Center>

                        <Icon mb="1" as={<MaterialCommunityIcons name="account"/>} color="white" size="sm" onPress={() => props.navigation.navigate('Profile')}/>

                        <Text color="white" fontSize="12">

                            Профиль

                        </Text>

                        </Center>

                    </Pressable>

                    </HStack>

                </Box>

            </NativeBaseProvider>

            }

        </>

    )

}

const styles = StyleSheet.create({

    container: {

      flex: 1,

      backgroundColor: '#fff',

      justifyContent: 'center',

    },

  });