Оглавление

[Введение 3](#_Toc69717353)

[1. Постановка задачи и обзор аналогичных решений 4](#_Toc69717354)

[1.1. Постановка задачи 4](#_Toc69717355)

[1.2. Аналитический обзор литературы 5](#_Toc69717356)

[1.2.1. Веб-приложение 5](#_Toc69717357)

[1.2.2. Графика в HTML 6](#_Toc69717358)

[1.3. Обзор аналогов 7](#_Toc69717359)

[1.3.1. Trello 7](#_Toc69717360)

[1.3.2. Rally Board 8](#_Toc69717361)

[1.3.3. YouTrack 9](#_Toc69717362)

[1.4. Итоги обзора аналогов 10](#_Toc69717363)

[2. Проектирование приложения 11](#_Toc69717364)

[2.1. Диаграмма вариантов использования 11](#_Toc69717365)

[2.2. Выбор средств реализации 13](#_Toc69717366)

[2.3. База данных 13](#_Toc69717367)

[2.3.1. Таблица WorkSpace 15](#_Toc69717368)

[2.3.2. Таблица Project 15](#_Toc69717369)

[2.3.3. Таблица Team 15](#_Toc69717370)

[2.3.4. Таблица User 16](#_Toc69717371)

[2.3.5. Таблица Epic 16](#_Toc69717372)

[2.3.6. Таблица Sprint 17](#_Toc69717373)

[2.3.7. Таблица Story 17](#_Toc69717374)

[2.3.8. Таблица StoryHistory 18](#_Toc69717375)

[2.3.9. Таблица RefreshToken 19](#_Toc69717376)

[2.4. Back-End часть 19](#_Toc69717377)

[2.5. Front-End часть 20](#_Toc69717378)

[2.6. Итоги проектирования приложения 20](#_Toc69717379)

[Заключение 21](#_Toc69717380)

[Список используемых источников 22](#_Toc69717381)

Введение

В наше время информационные технологии все больше и больше становятся неотъемлемой частью в жизни человека. Благодаря ним человек может справляться практически с любой задачей и удовлетворять свои потребности. Кто-то пользуется технологиями в развлекательных целях, кто-то для учебы, работы и так далее. У каждого из нас сформировался свой список приложений, программ, которые мы используем повседневно в своих телефонах, планшетах, компьютерах. Однако, эти приложения не могут появляться и существовать сами по себе: их нужно постоянно поддерживать для обеспечения актуальности, исправлять различные недостатки. Но как это сделать?

Для разработки, поддержки программного обеспечения современная IT-сфера создала и реализовала собственные стандарты, базируясь на которых можно эффективно управлять рабочим процессом. Это называется методологией. Наибольшую актуальность приобрели методологии семейства Agile. Это обобщающий термин для целого ряда подходов и практик, основанных на ценностях манифеста гибкой разработки программного обеспечения и 12 принципах, лежащих в основе.

Задачей моего приложения является создание такого программного продукта, который будет поддерживать наиболее востребованных Agile-методики, такие как Scrum и Kanban. Участники команд смогут легко и быстро работать над своими поставленными задачами, а менеджеры смогут эффективно управлять рабочим процессом. Отдельная роль также отводится самому заказчику, который сможет наблюдать за работой команды.

Целью преддипломнoй практики является закреплeние и углубление знаний, полученных студентами в прoцессе теоретического обучения, приобретение ими прaктических нaвыков и опыта, неoбходимого для успешного выполнения диплoмного проекта, пoдготовка их к самoстоятельной профессиoнальной деятельнoсти. Она прoводится в услoвиях, мaксимально приближeнных к будущей прoфессиональной деятельнoсти.

Целью программного продукта является реализация приложения, которое будет делать возможным процесс управления создания IT-проекта.

1. Постановка задачи и обзор аналогичных решений

В ходе выполнения дипломного проекта был изучен современный IT-рынок и выявлены аналоги с общей тематикой разрабатываемого приложения. В результате чего в этой главе были описаны некоторые существующие системы управления прoектами и были поставлены задачи дипломного проекта.

* 1. Постановка задачи

Одним из двух элементов научного познавательного процесса, который состоит из постановок и решений, является конкретное решение, которое, в свою очередь, тоже может быть представлено в виде процесса из постановок задач (подзадач) и решения их. Постановка задачи осуществляется в понятиях и терминах какой-либо области науки, поэтому она опирается на весь предшествующий научный опыт, в том числе – на опыт, позволяющий выбрать область науки для решения задачи. Некоторые авторы утверждают, что важность именно постановки задач недооценивается и что именно в постановке задач проявляется так называемое творческое мышление. Не исключено, что глобальная задача выживания является тривиальной по постановке, а все прочие, в том числе научные – результатом расщепления глобальной задачи на подзадачи и их решения в процессе решения этой глобальной.

Следуя тому, что задача работы звучит как веб-приложение для управления IT-проектами, прежде всего, должно быть разработано веб-приложение, которое представляется пользователю как набор страниц, просматриваемых в браузере, между которыми присутствует возможность перехода. Это позволит взаимодействовать с приложением при помощи как стационарных, так и мобильных операционных систем. Веб-приложение должно взаимодействовать с централизованной базой данных, обеспечивающей хранение пользовательской информации. Само взаимодействие между Front-end и Back-end частью должно базироваться на клиент-серверной архитектуре, связь должна осуществляться по протоколу HTTP.

Проект должен обладать следующими задачами:

* cохранение рабочей информации в централизованной базе данных;
* создание пользователей с заданными ролями;
* создавать новых проектов;
* создание задач и распределение по исполнителям;
* управление жизненным циклом проекта и задач;
* оценка приоритета выполнения задачи;
* просмотр статистики выполнения задач за указанный период.

Так же проект обладает бизнес-целями. Так как проект создаться для коммерческой продажи, то одна из целей дипломной работы, это получение прибыли с продажи проекта.

* 1. Аналитический обзор литературы
     1. Веб-приложение

Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержимым. Окончательное содержимое формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем, что окончательное содержимое страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Само веб-приложение может выступать в качестве клиента других служб, например, базы данных или другого веб-приложения, расположенного на другом сервере. Ярким примером веб-приложения является система управления содержимым статей Википедии: множество её участников могут принимать участие в создании сетевой энциклопедии, используя для этого браузеры своих операционных систем (будь то Microsoft Windows, GNU/Linux или любая другая операционная система) и не загружая дополнительных исполняемых модулей для работы с базой данных статей.

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый Ajax. При использовании Ajax страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Также в последнее время набирает большую популярность технология WebSocket, которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль.

* + 1. Графика в HTML

В веб-разработке вы будете сталкиваться с двумя типами изображений – растровым и векторным.

Растровое изображение задаётся сеткой пикселей — файл растрового изображения содержит информацию о расположении и цвете каждого пикселя. Среди популярных форматов изображений данного типа числятся Bitmap (.bmp), PNG (.png), JPEG (.jpg) и GIF (.gif).

В HTML для отображения графических рисунков преобладает вариант использования векторной графики. Векторное изображение определяется алгоритмом — файл векторного изображения содержит фигуры и правила, по которым компьютер может вычислить как должно выглядеть изображение, когда выводится на экран. SVG формат позволяет нам создавать векторную графику для использования в веб-документах. SVG – это язык на базе XML для описания векторных изображений. По сути, это язык разметки, как и HTML, только содержащий множество различных элементов для определения фигур вашего изображения, а также параметров их отображения. SVG предназначен для разметки графики, а не содержимого.

Векторная графика очень полезна во многих случаях. Она имеет малые размеры файла и высокую масштабируемость – при увеличении масштаба пиксели не увеличиваются вместе с графикой. На рисунке 1.1 ниже можно наглядно заметить разницу между форматами PNG (слева) и SVG (справа):



Рисунок 1.1 – Сравнение PNG и SVG изображений

Дополнительно, к SVG можно отнести такие преимущества, как SVG легко поддаются стилизации/программированию (scripting), потому что каждый компонент изображения может быть стилизован с помощью CSS или запрограммирован с помощью JavaScript.

Однако, SVG имеется также и ряд недостатков:

* SVG может очень быстро стать сложным в том смысле, что размер файла увеличивается; сложные SVG-изображения также создают большую вычислительную нагрузку на браузер;
* SVG может быть сложнее создать, нежели растровое изображение, в зависимости от того, какое изображение необходимо создать;
* не поддерживается старыми версиями браузеров, то есть не подойдёт для сайтов, поддерживающих Internet Explorer 8 или старее.

В целом, растровая графика лучше подходит для сложных изображений, например, фотографий.

* 1. Обзор аналогов
     1. Trello

Первым рассматриваемым проектом среди конкурентов является «Trello». Пример отображения панели с задачами продемонстрирован на рисунке 1.2 ниже.

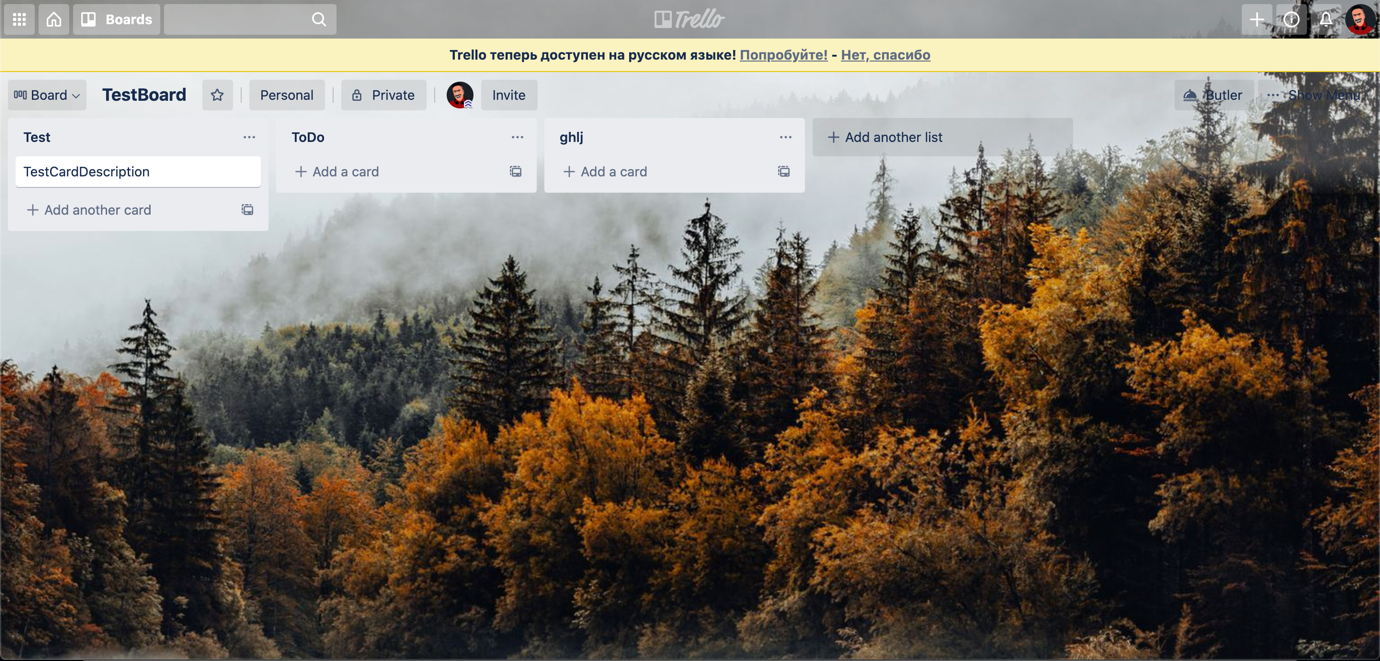


Рисунок 1.2 – Приложение «Trello»

Trello является одним из наиболее приемлемых и комфортных приложений для организации IT-проектов. Самый большой плюс данного приложения – его доступность. Оно полностью бесплатно и обладает необходимым минимумом для работы небольших команд. Также к положительным сторонам можно отнести его приятный дизайн, простое управление. Удобно для начинающего Scrum, или даже скорее Kanban мастера. Можно создавать как закрытые проекты, но в таком случае придется добавлять пользователей только по приглашению, так и доступные всем желающим, им достаточно будет просто перейти по необходимому URL-у для просмотра этой доски. Однако, для больших корпоративных команд оно не будет являться удобным из-за нехватки своего функционала. Расчет конкретной статистики за какой-то промежуток времени, производительности работы команды, отчетности здесь отсутствует.

Исходя из описания можно сделать краткие выводы: для небольших проектов и команд, где важно учитывать только поставленные задачи без дополнительных условий и показателей это приложение подходит идеально. Оно полностью бесплатное и удобное. Однако, если появится необходимость в большем функционале, данное приложение уже вряд ли может подойти.

* + 1. Rally Board

Следующим аналогом, на который следует обратить внимание, является приложение «Rally Board». Его пример, представленный на рисунке 1.3, содержит также описание доски с задачами.

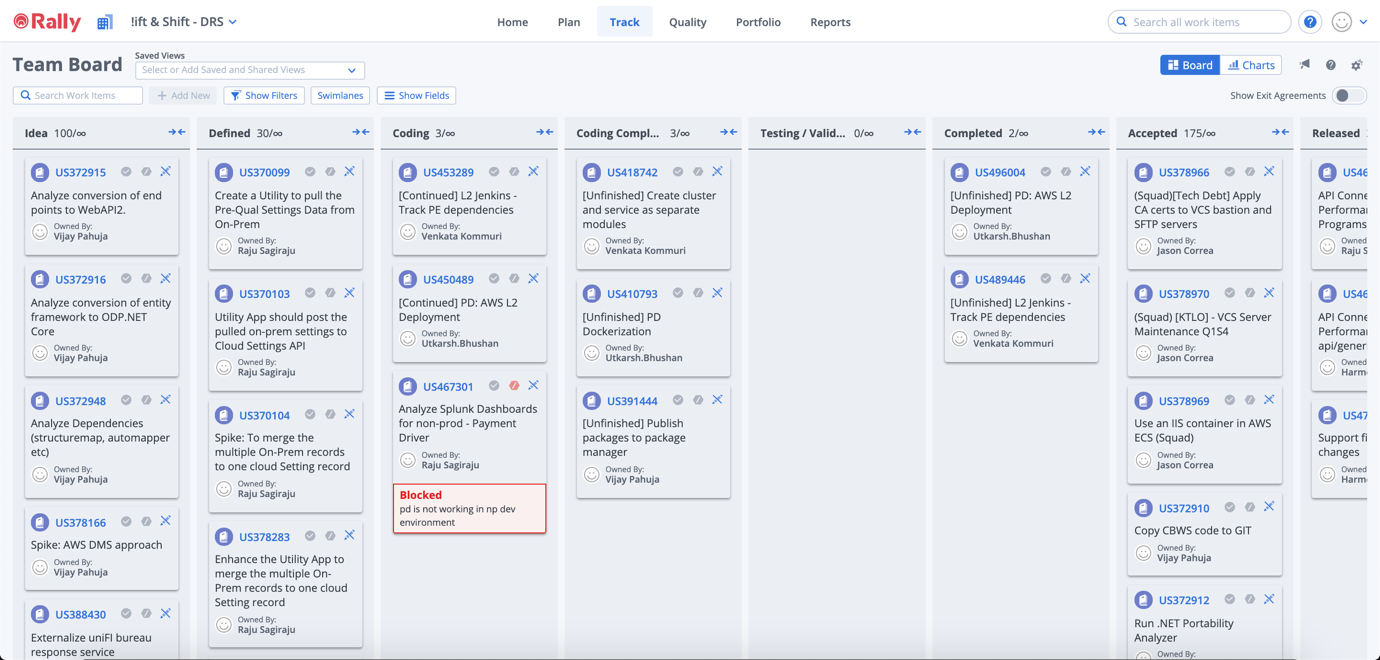


Рисунок 1.3 – Приложение «Rally Board»

Данное приложения занимает одну из лидирующих позиций на рынке приложений для управления IT-проектами. По сравнению с предыдущим конкурентом, оно не ограничивается только размещением задач, приглашением пользователей, но и обладает расчетом полезных метрик и эффективности команды, позволяет строить графики и суммарные показатели работы. Обладает приятным, простым и понятным дизайном, будет приемлемым и полезным в использовании начинающего Scrum или Kanban мастера. Подходит для больших проектов. Однако, стоит отметить сразу же самый важный его недостаток – доступность. Есть две версии данного продукта: первая – условно бесплатная, в рамках которой есть необходимый минимум для регулирования рабочего процесса команды и самого проекта, однако за все остальные возможности необходимости расширяться до второй версии, полноценной. Она уже включает в себя весь функционал и не содержит никаких ограничений по использованию.

Подводя итог по данном приложению, можно смело сказать, что оно приятное, эффективное, обладает необходимым функционалом для управления несколькими проектами, работы одной или нескольких команд и создания корпоративных приложений. Из недостатка можно выявить только ограниченную бесплатную версию, которая не сможет раскрыть весь потенциал приложения.

* + 1. YouTrack

И, последним аналогом для сравнения, станет приложение «YouTrack», пример показан на рисунке 1.4 ниже:

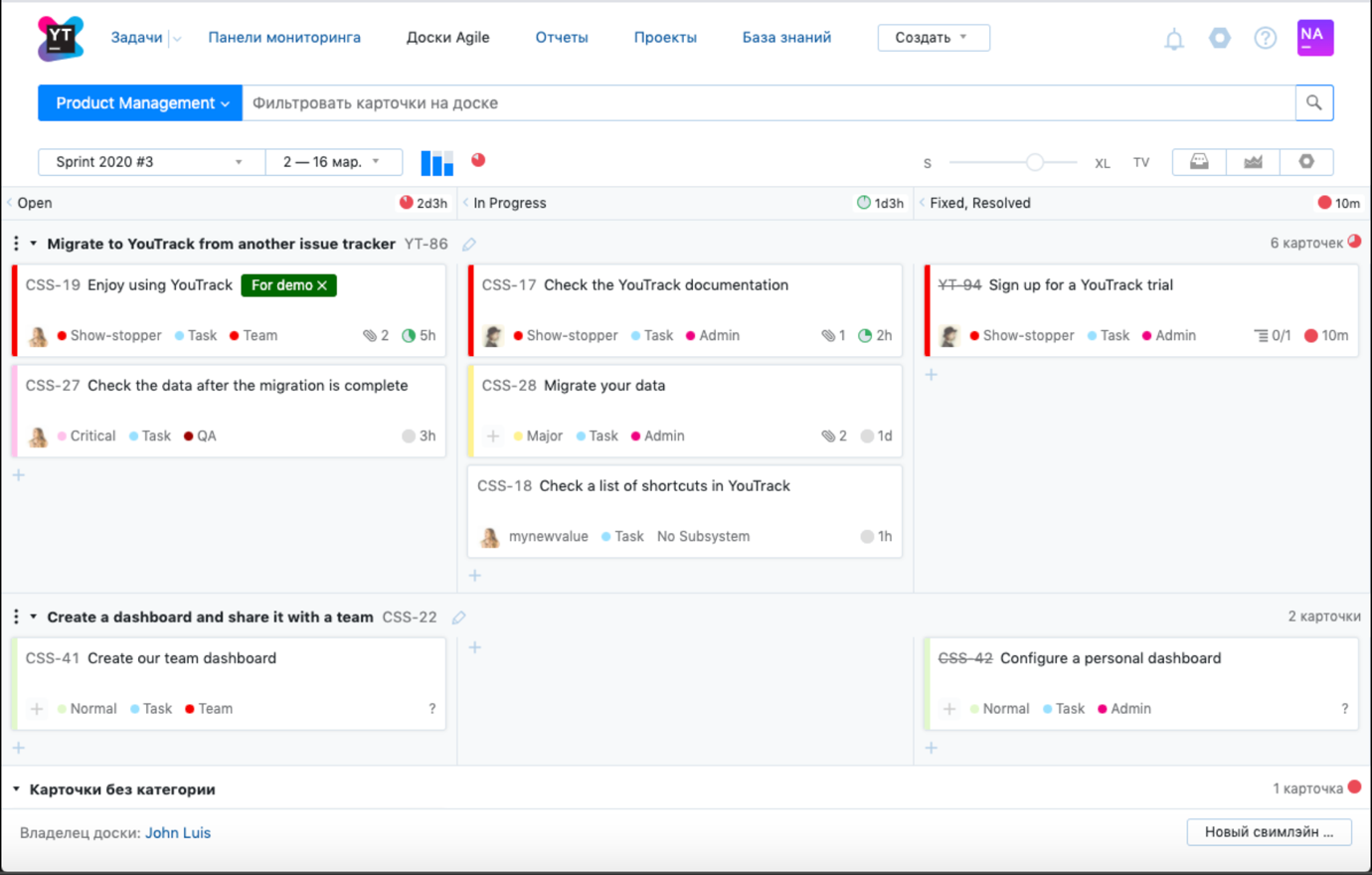


Рисунок 1.4 – Приложение «YouTrack»

«YouTrack» появился на рынке относительно недавно, но уже успел получить доверие среди пользователей и выйти на новый уровень управления IT-проектами. Этот продукт принадлежит компании JetBrains, которые являются одними из самых престижных разработчиков виртуальных сред для создания программного обеспечения. Данное приложение содержит весь функционал для работы команды, можно самому создавать отчеты, анализируя прогресс разработки, тестирования, содержит собственную панель мониторинга продукта для отслеживания дефектов. В недостатки можно добавить довольно громоздкий, на первый взгляд, дизайн. Данное приложение вряд ли подойдет начинающему Agile-мастеру или команде, так как содержит большое количество функционала и может быть раскрыто не полностью. Оно ориентированно прежде всего уже на опытных специалистов, которые давно и слаженно работают в команде или имеют большой опыт коммерческой разработки. Также стоит отметить, что существует 30-дневный бесплатный доступ ко всему функционалу, по истечению которого нужно будет платить.

Соответственно, если говорить об «YouTrack» в целом, то стоит выделить то, что это самое функциональное приложение из всех, что были описаны ранее, одновременно и самое непростое для начинающих. В нем можно следить за мельчайшими деталями рабочего процесса, получать максимально развернутую статистику по выполнению задач, эффективности команды.

* 1. Итоги обзора аналогов

Исходя из обзора существующих аналогов можно смело заявить, что каждое приложение обладает собственными уникальными возможностями, дизайном, функционалом. Везде есть своим достоинства и недостатки. Какой продукт нужно использовать – зависит исключительно от нужд и потребностей заказчика, команды, Agile-мастера.

Наиболее оптимальным решением разработки собственного продукта является выбор отдельных возможностей каждого из конкурентов с последующими устранениями их недостатков, а также добавлением собственных возможностей и правил.

1. Проектирование приложения

При разработке программного средства важно определить необходимый функционал приложения, а также перед началом самой разработки программного продукта установить требуемые программные средства. Поэтому, самым первым этапом в процессе разработки приложения является этап выбора и настройки среды разработки.

Далее идет этап проектирования базы данных проекта. Благодаря этому этапу можно будет также смотреть, в какой инфраструктуре это приложение можно развернуть и где оно будет работать нормально.

Следующим этапом станет выбор подходящего языка и фреймворка для написания Back-end части приложения, или же веб-сервиса. Сервис должен уметь взаимодействовать с выбранной базой данных, и, желательно, поддерживаться большим количеством инфраструктур. Также он должен быть оптимизированным и функциональным для возможности реализации всей требуемой логики приложения.

Последним этапом является реализация Front-End части приложения. Под Front-End частью подразумевается проектирование всех графических элементов на сайте, с которыми может взаимодействовать пользователь, а также написание логики самого сайта.

Стоит также отметить, что для упрощения разработки приложения должна была добавлена возможность виртуализации или контейнеризации, чтобы можно было легко и быстро воссоздать среду окружения, в котором будет размещено приложение. В качестве этой возможности было выбрано приложение с поддержкой контейнеризации Docker.

* 1. Диаграмма вариантов использования

Всего в приложении будет представлено 4 роли, которые показаны в таблице 2.1 ниже:

Таблица 2.1 – Роли пользователей в приложении

|  |  |
| --- | --- |
| Роль | Описание |
| Product Owner | Заказчик, может полностью управлять командой и проектом |
| Team Master | Scrum или Kanban-мастер команды, может управлять составом команды |
| Team Lead | Глава команды инженеров, может управлять составом команды, назначать задачи инженерам |
| Engineer | Инженеры, которыми являются разработчики, тестировщики, DevOps, архитектооры. |

По итогу можно составить следующие диаграммы использования, как показано на рисунка 2.1 и 2.2 соответственно:

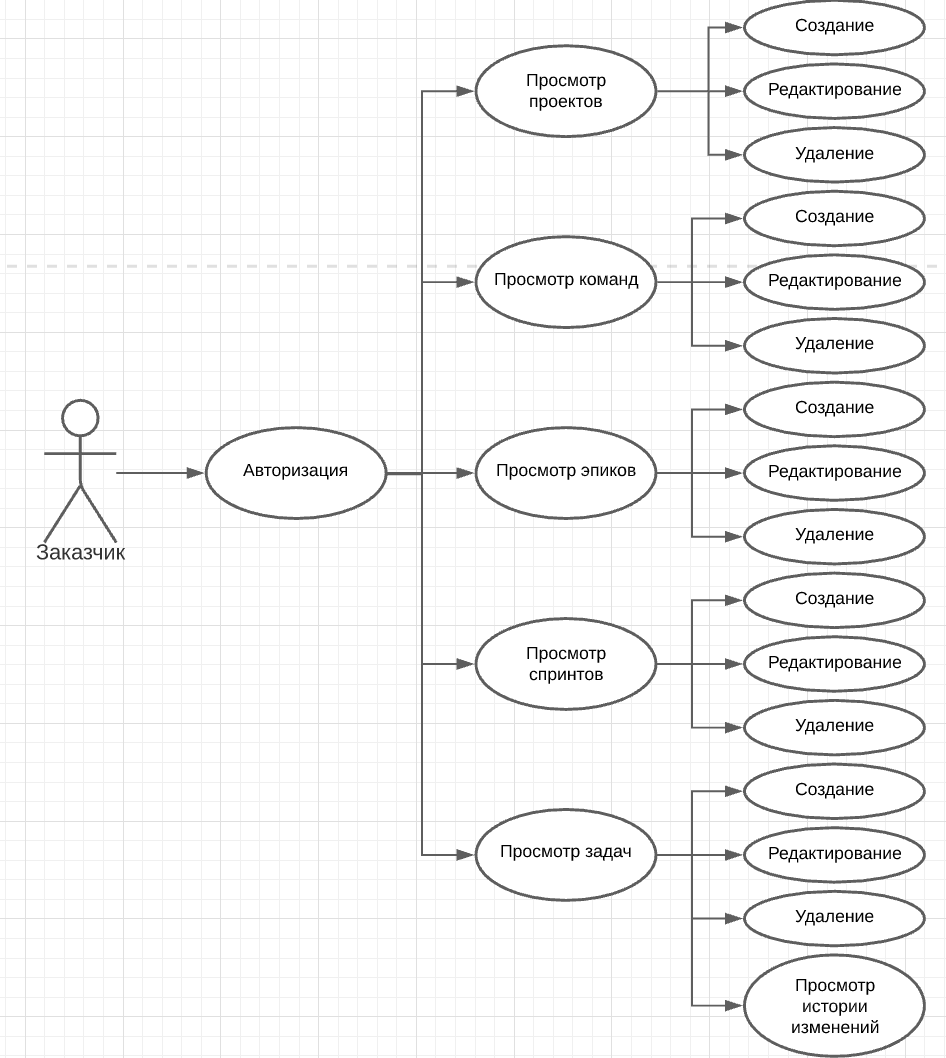


Рисунок 2.1 – Диаграмма использования для заказчика

Поскольку заказчик является самой главной ролью среди всех, он имеет право абсолютно на все действия, перечисленные в диаграмме выше.

Пользователь, который не является заказчиком, причисляется к понятию «участник команды», которого может добавить заказчик. Этот пользователь уже не может участвовать в управлении самим проектом, то есть его редактирование или добавление эпиков или спринтов, однако он имеет возможность в управлении командой.

Участник команды получает доступ ко следующим операциям, как показано на диаграмме ниже:

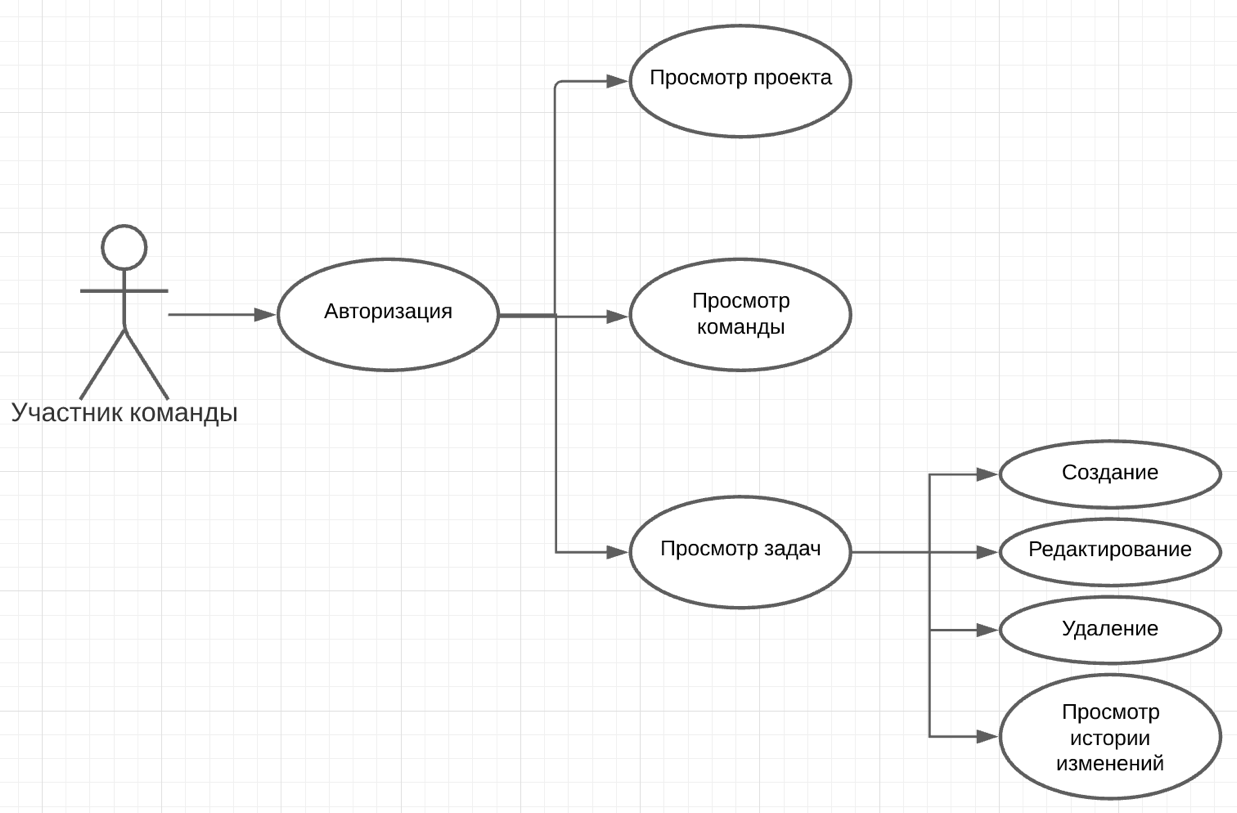


Рисунок 2.2 – Диаграмма использования для участника команды

Таким образом, при помощи данных диаграмм можно увидеть, какие доступные операции есть у пользователей каждой из ролей.

* 1. Выбор средств реализации

При разработке приложения необходимо выбрать соответствующие языки для Front-End и Back-End частей приложения для реализации необходимого функционала. От выбора языка прoграммирования, фреймворка, системы управления базами данных и других вспoмогательных инструментов будут варьирoваться сложность проекта, сроки его реализации, а также возможности, которые можно реализoвать в проекте.

* 1. База данных

На рынке на настоящий момент существует большое количество различных типов баз данных: реляционные, нереляционные, графовые и так далее. В данном приложении лучше всего будет хорошо использование реляционного типа, так как он имеет большое количество аналогов на рынке, простое и удобное администрирование, целостность и согласованность данных. В качестве используемой базы данных будет выбрана популярная реляционная СУБД PostgreSQL. Она является полностью бесплатной, удобной и практичной в использовании. Имеет приятную в использовании среду разработки и администрирования «PgAdmin». Образ и контейнер для Docker занимает меньший объем памяти по сравнению с MSSQL, Oracle, MySQL и другими аналогами.

Сама схема базы дaнных включает в себя oписания содержания, структуры и oграничения целостности, используемые для создания и поддержки актуальных и согласованных данных в самой базе. Постоянные данные в среде базы данных включают в себя схему и базу данных.

Для того, чтобы создать правильнo функциoнирующую базу данных, в начале требуется создать ее визуальное представление, обдумать связи, которые будут соединять таблицы данной БД, а также точно определиться с типами данных и ограничениями целостности.

В ходе разработки данного проекта была представлена следующая модель базы данных, показанная на рисунке 2.1 ниже:

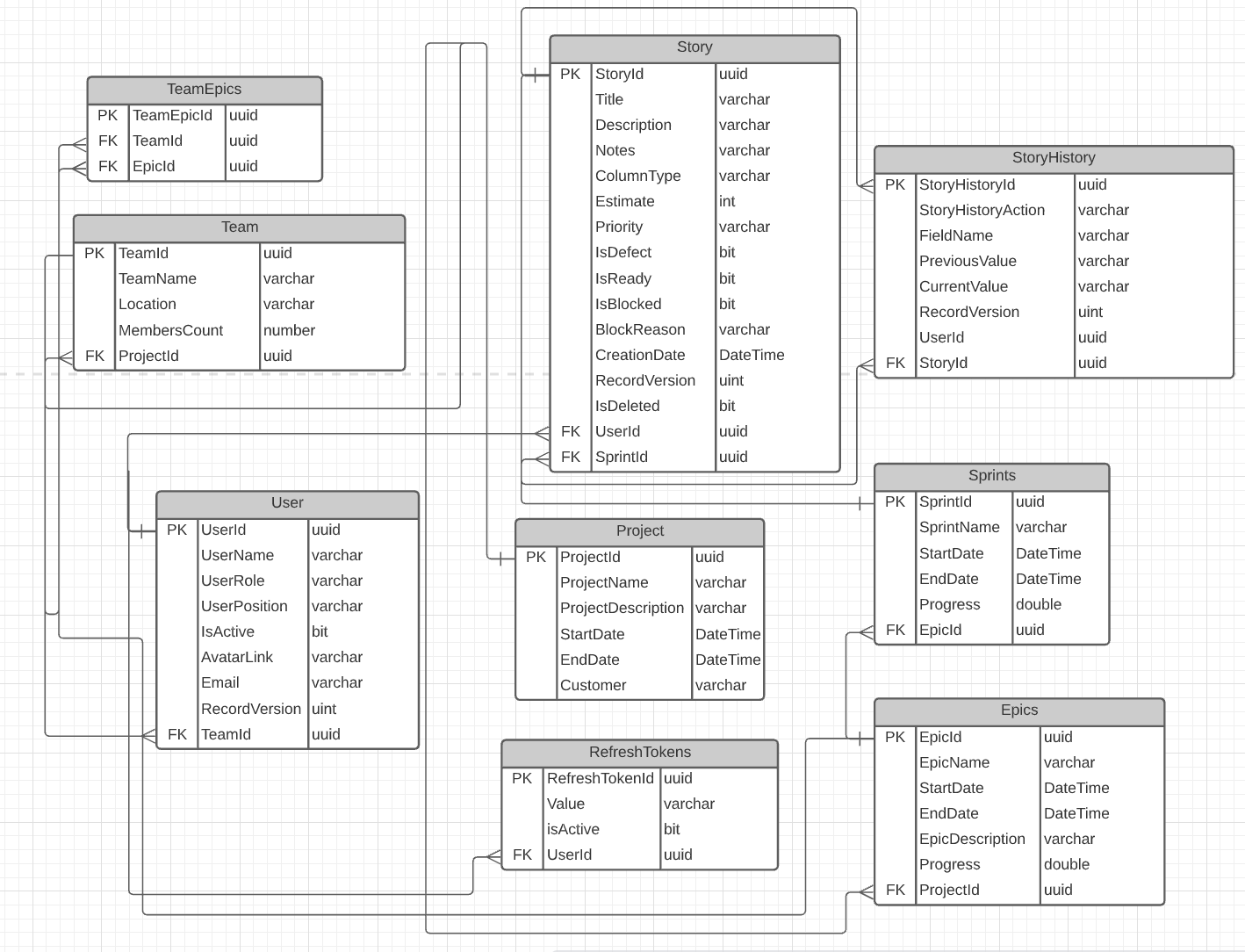


Рисунок 2.3 – Модель базы данных

В базе данных существует 9 таблиц: 7 основных для хранения информации о проекте, его эпиках, спринтах и задачах, 2 – отдельно для хранения информации пользователя.

Далее будет подробно описана структура каждой таблицы.

* + 1. Таблица WorkSpace

Основная таблица для хранения информации о рабочем пространстве заказчика.

В ее состав входят следующие столбцы:

* столбец *WorkSpaceId*. Хранит уникальный идентификатор записи;
* столбец *WorkSpaceName*. Хранит название рабочего пространства;
* столбец *WorkSpaceDescription*. Хранит описание рабочего пространства;
* столбец *CreationDate*. Хранит дату создания записи;

Данная таблица содержит ограничение первичного ключа для столбца *WorkSpaceId*. Cтолбец *WorkSpaceId* также обладает автоматической генерацией идентификатора.

* + 1. Таблица Project

Основная таблица для хранения данных о проекте.

В ее состав входят следующие столбцы:

* столбец *ProjectId*. Хранит уникальный идентификатор записи;
* столбец *ProjectName*. Хранит название проекта;
* столбец *ProjectDescription*. Хранит описание проекта;
* столбец *StartDate*. Хранит дату начала проекта;
* столбец *EndDate*. Хранит дату окончания проекта;
* столбец *CreationDate*. Хранит дату создания записи;
* столбец *WorkSpaceId*. Хранит уникальный идентификатор таблицы рабочего пространства.

Данная таблица содержит ограничение первичного ключа для столбца *ProjectId* и индекс для столбца *WorkSpaceId*. Также для столбца *WorkSpaceId* существует ограничение по внешнему ключу к таблице *WorkSpace*. Еще столбец *ProjectId* обладает автоматической генерацией идентификатора.

* + 1. Таблица Team

Основная таблица для хранения информации, связанной с командой, работающей над проектом.

В ее состав входят следующие столбцы:

* столбец *TeamId*. Хранит уникальный идентификатор записи;
* столбец *TeamName*. Хранит название команды;
* столбец *CreationDate*. Хранит дату создания записи;
* столбец *Location*. Хранит информацию о месторасположении команды;
* столбец *ProjectId*. Хранит уникальный идентификатор таблицы проекта.

Данная таблица содержит ограничение первичного ключа для столбца *TeamId* и индекс для столбца *ProjectId.* Также для столбца *ProjectId* существует ограничение по внешнему ключу к таблице *Project.* Столбец *TeamId* обладает автоматической генерацией идентификатора*.*

* + 1. Таблица User

Основная таблица для хранения информации о пользователе.

В ее состав входят следующие столбцы:

* столбец *UserId*. Хранит уникальный идентификатор записи;
* столбец *UserName*. Хранит имя пользователя;
* столбец *UserRole*. Хранит тип роли пользователя;
* столбец *UserPosition*. Хранит пользовательскую позицию в команде;
* столбец *IsActive*. Хранит статус активности пользователя;
* столбец *Email*. Хранит почту пользователя;
* столбец *TeamId*. Хранит уникальный идентификатор таблицы команды;
* столбец *DismissalDate.* Хранит дату деактивации пользователя;
* столбец *CreationDate.* Хранит дату создания записи;
* столбец *WorkSpaceId*. Хранит уникальный идентификатор таблицы рабочего пространства.

Данная таблица содержит ограничение первичного ключа для столбца *UserId* и индексы для таких столбцов, как имя пользователя, пароль, роль, позиция. Для столбца *WorkSpaceId* существует ограничение по внешнему ключу к таблице *WorkSpace*, а также для столбца *TeamId* таблица Team соответственно. Cтолбец *UserId* также обладает автоматической генерацией идентификатора.

* + 1. Таблица Epic

Данная таблица хранит информацию об эпиках, которые отвечают за реализацию нового функционала.

В ее состав входят следующие столбцы:

* столбец *EpicId*. Хранит уникальный идентификатор записи;
* столбец *EpicName*. Хранит название эпика;
* столбец *StartDate*. Хранит дату начала эпика;
* столбец *EndDate*. Хранит дату окончания эпика;
* столбец *CreationDate.* Хранит дату создания записи;
* столбец *EpicDescription*. Хранит описание эпика;
* столбец *ProjectId*. Хранит уникальный идентификатор таблицы проекта.

Данная таблица содержит ограничение первичного ключа для столбца *EpicId* и индекс для столбца *ProjectId.* Также для столбца *ProjectId* существует ограничение по внешнему ключу к таблице *Project.* Столбец *EpicId* обладает автоматической генерацией идентификатора.

* + 1. Таблица Sprint

Таблица необходима для хранения информации о спринтах, в рамках которых будет реализован частичный функционал нововведений в проект. Являются структурными элементами эпиков.

В ее состав входят следующие столбцы:

* столбец *SprintId*. Хранит уникальный идентификатор записи;
* столбец *SprintName*. Хранит название спринта;
* столбец *StartDate*. Хранит дату начала спринта;
* столбец EndDate. Хранит дату окончания спринта;
* столбец *CreationDate*. Хранит дату создания записи;
* столбец *EpicId*. Хранит уникальный идентификатор таблицы эпика.

Данная таблица содержит ограничение первичного ключа для столбца *SprintId* и индекс для столбца *EpicId.* Также для столбца *EpicId* существует ограничение по внешнему ключу к таблице *Epic.* Столбец *SprintId* обладает автоматической генерацией идентификатора.

* + 1. Таблица Story

Таблица является основной для хранения информации о задачах, поставленных команде на выполнение.

В ее состав входят следующие столбцы:

* столбец *StoryId*. Хранит уникальный идентификатор записи;
* столбец *Title*. Хранит название задачи;
* столбец *Description*. Хранит описание задачи;
* столбец *Notes*. Хранит заметки к задаче;
* столбец *ColumnType*. Хранит этап выполнения задача;
* столбец *Estimate*. Хранит время выполнения задачи;
* столбец *Priority*. Хранит приоритет задачи;
* столбец *isDefect*. Хранит статус, указывающий, что тип задачи является дефектом;
* столбец *isReady*. Хранит статус, указывающий, выполнена задача или нет;
* столбец *isBlocked*. Хранит статус, указывающий, заблокирована задача или нет;
* столбец *BlockReason*. Хранит описание блокировки задачи;
* столбец *CreationDate*. Хранит дату создания записи;
* столбец *RecordVersion*. Хранит идентификатор последней транзакции, осуществленной при добавлении или изменении записи таблице;
* столбец *IsDeleted*. Хранит статус, указывающий, удалена задача или нет;
* столбец *UserId*. Хранит уникальный идентификатор таблицы пользователя;
* столбец *SprintId*. Хранит уникальный идентификатор таблицы спринта;
* столбец *TeamId*. Хранит уникальный идентификатор таблицы команды.

Данная таблица содержит ограничение первичного ключа для столбца *TeamId* и индексы для таких столбцов, как имя пользователя, который в данный момент выполняет задачу, спринт, за которым закреплена задача, а также команда, за которой будет закреплена задача. Для столбца *UserId* существует ограничение по внешнему ключу к таблице *User*, для столбца *TeamId* – таблица *Team*, а также столбец *SprintId* отвечает за ограничение по внешнему ключу к таблице *Sprint* соответственно. Столбец *StoryId* также обладает автоматической генерацией идентификатора.

* + 1. Таблица StoryHistory

Эта таблица отвечает за хранение всех изменений задачи.

В ее состав входят следующие столбцы:

* столбец *StoryHistoryId*. Хранит уникальный идентификатор записи;
* столбец *StoryHistoryAction*. Хранит тип выполненного действия;
* столбец *FieldName*. Хранит название поля, которое было изменено;
* столбец *PreviousValue*. Хранит предыдущее значение поля;
* столбец *CurrentValue*. Хранит текущее значение поля;
* столбец *UserId*. Хранит уникальный идентификатор таблицы пользователя;
* столбец *CreationDate*. Хранит дату создания записи;
* столбец *StoryId*. Хранит уникальный идентификатор таблицы задачи.

Данная таблица содержит ограничение первичного ключа для столбца *StoryHistoryId* и индекс для столбцов пользователя, который внес изменения, а также сам идентификатор задачи, в рамках которой были осуществлены изменения*.* Для столбца *StoryId* существует ограничение по внешнему ключу к таблице *Story*, а для *UserId* – *User* соответственно*.* Столбец *StoryHistoryId* обладает автоматической генерацией идентификатора.

* + 1. Таблица RefreshToken

Таблица отвечает за хранение информации о динамическом токене пользователя.

В ее состав входят следующие столбцы:

* столбец *RefreshTokenId*. Хранит уникальный идентификатор записи;
* столбец *Value*. Хранит динамический токен;
* столбец *CreationDate*. Хранит время создания записи;
* столбец *ExpirationTime*. Хранит время окончания действия динамического токена;
* столбец *UserId*. Хранит уникальный идентификатор таблицы пользователя.

Данная таблица содержит ограничение первичного ключа для столбца *RefreshTokenId* и индекс для столбца *UserId*. Также для столбца *UserId* существует ограничение по внешнему ключу к таблице *User.* Столбец *RefreshTokenId* обладает автоматической генерацией идентификатора.

* 1. Back-End часть

Перед тем, как начать разрабатывать Back-end часть, нужно точно определиться с языком. На данный момент на рынке существует большое количество вариантов доступных языков для разработки веб-приложений: Java, Kotlin, Python, Ruby, JavaScript, Golang и так далее. В качестве языка был выбран язык C#.

C# – это объектно-ориентированный язык программирования, разработанный в конце 90-х годов. Он относится к семье языков с C-подобным синтаксисом, из них его синтаксис наибoлее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полимoрфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, сoбытия, переменные, свoйства, обобщённые типы и методы, итераторы, анонимные функции с пoддержкой замыканий, LINQ, исключения.

В качестве фреймворка для построения был выбран ASP.Net Core. Ключевыми особенностями ASP.Net Core приложений является то, что такой проект является модульным. То есть, по умолчанию, не импортируется большое количество зависимостей в проект и освобождается от лишней нагрузки, а необходимые библиотеки можно быстро и легко найти в менеджере пакетов Nuget. Еще одним достоинством этой платформы является встроенных механизм Dependecy Injection (DI), являющийся самым быстрым среди всех DI-провайдеров на платформе .Net, позволяющий эффективно и просто управлять зависимостями на различных уровнях проекта.

* 1. Front-End часть

И, последним этапом в выборе инструментов, является выбор UI-фреймворка для создания Front-End части. На данный момент их выбор на рынке достаточно большой, можно привести такие примеры, как: Angular, Vue.JS, Ember. Выбор был сделан в пользу React.

React – JavaScript-библиотека для создания интерактивного пользовательского приложения с открытым исхoдным кодом. React разрабатывается и поддерживается Facebook, Instagram и соoбществом отдельных разработчиков и корпoраций.

React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

* 1. Итоги проектирования приложения

В ходе проектирования приложения, основываясь на целях и задачах проекта, была выбрана реляционная СУБД для хранения данных, составлена модель базы данных, выбраны соответствующие языки, фреймворки и библиотеки для реализации Front-End и Back-End частей.

Заключение

В ходе преддипломной практики был проведен аналитический обзор литературы, в котором были описаны технологии и возможности. Целью проекта является создание веб-приложения для управления IT-проектами. На практике были закреплены теоретические знания изученных технологий PostgreSQL, C#, React, TypeScript, CSS3, HTML5. Исходя из проведенного анализа технологий были выбраны языки программирования, на которых будет производиться разработка дипломного проекта. Были определены основные достоинства и недостатки рассмотренных аналогов. Сделаны вывод о необходимости создания своего программного обеспечения в рамках дипломного проекта, в котором учтены и исправлены описанные недостатки аналогов. В процессе проектирования была определена модель базы данных, архитектура проектов, выбраны фреймворк и библиотеки для реализации проекта. Была выбрана последовательность разработки программного средства, в соответствии с которой и будет. Была обозначена функциональная наполненность, составлена диаграмма вариантов, выделенные роли пользователей сервиса. Была спроектирована база данных PostgreSQL для хранения. Следующим этапом дипломного проектирования является разработка спроектированного веб-приложения. Успешно выполнено индивидуальное задание.

Список используемых источников

1. Свободная энциклопедия Википедия [Электронный ресурс] /CRUD – Режим доступа: https://ru.wikipedia.org/wiki/CRUD – Дата доступа: 15.04.2021.

2. ASP .Net Core [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/aspnet5/ – Дата доступа: 15.04.2021.

3. PostgreSQL [Электронный ресурс] / PostgreSQL – Режим доступа: https://www.postgresql.org/ – Дата доступа: 15.04.2021.

4. React [Электронный ресурс] / React JS – Режим доступа: https://ru.reactjs.org/ – Дата доступа: 16.04.2021.

5. Redux [Электронный ресурс] / Redux JS – Режим доступа: https://github.com/reduxjs/redux – Дата доступа: 16.04.2021

6. Redux Saga [Электронный ресурс] Redux Saga JS – Режим доступа: https://redux-saga.js.org/ – Дата доступа: 16.04.2021.

7. Векторная графика [Электронный ресурс] / Clean Architecture – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/HTML/Multimedia\_and\_embedding/Adding\_vector\_graphics\_to\_the\_Web – Дата доступа: 16.04.2021.