Реферат

Пояснительная записка дипломного проекта содержит 118 страниц пояснительной записки, 10 таблиц, 15 формул, 56 иллюстраций, 15 источников литературы, 12 приложений.

Веб-приложение, REACT, ASP.NET, TYPESCRIPT, C#, MATERIAL UI, POSTRGRESQL, DOCKER, REDIS.

Целью дипломного проекта является разработка веб-приложения для управления IT-проектами.

В первой главе проводится аналитический обзор литературы по тематике дипломного проекта.

Вторая глава посвящена обзору средств разработки и содержит описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается контрольный пример, с проведением тестирования, и показывается поведения системы при разных внештатных ситуациях.

В пятой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе приводится расчет экономических параметров и себестоимости программного средства.

Объем графической части дипломного проекта составляет 1.75 листа А1.

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

Abstract

Explanatory note of the diploma project 118 pages of explanatory note, 7 tables, 15 formulas, 51 illustrations, 18 sources of literature, 15 appendices.

WEB APPLICATION, REACT, ASP.NET, TYPESCRIPT, C#, MATERIAL UI, POSTRGRESQL, DOCKER, REDIS.

The aim of the diploma project is to develop a web application for IT-projects management.

The first chapter provides an analytical review of the literature on the subject of the diploma project.

The second chapter is devoted to a review of development tools and describes the technologies used during the project.

The third chapter describes the development process, the principles of functioning and the purpose of the created project components.

The fourth chapter describes a test case, with testing, as well as a demonstration of the behavior of the system in various emergency situations.

The fifth chapter describes a user manual that allows you to understand in detail the interface of the software tool.

The sixth chapter provides the calculation of economic parameters and the cost of software developed in the framework of the graduation project.

The volume of the graphic part of the diploma project is 1.75 sheets A1.

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

Оглавление

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

[Реферат 1](file:////Applications/TestProjects/Diploma/docs/Yaniuk_Diploma.docx#_Toc71657068)

[Abstract 2](file:////Applications/TestProjects/Diploma/docs/Yaniuk_Diploma.docx#_Toc71657069)

[Введение 5](file:////Applications/TestProjects/Diploma/docs/Yaniuk_Diploma.docx#_Toc71657070)

[1. Постановка задачи и обзор аналогичных решений 6](#_Toc71657071)

[1.1. Постановка задачи 6](#_Toc71657072)

[1.2. Обзор аналогичных решений 7](#_Toc71657073)

[1.2.1. Trello 7](#_Toc71657074)

[1.2.2. Rally Board 8](#_Toc71657075)

[1.2.3. YouTrack 9](#_Toc71657076)

[1.3 Патентный поиск 10](#_Toc71657077)

[2. Проектирование программного приложения 11](#_Toc71657078)

[2.1. Диаграмма вариантов использования 11](#_Toc71657079)

[2.2. Выбор средств реализации 12](#_Toc71657080)

[2.2.1. Основные языки программирования 13](#_Toc71657081)

[2.2.2. Фреймворки 14](#_Toc71657082)

[2.2.3. Система управления базами данных 15](#_Toc71657083)

[2.2.4. Контейнеризация 15](#_Toc71657084)

[2.2.5. Вспомогательные инструменты 16](#_Toc71657085)

[2.3. Проектирование базы данных 17](#_Toc71657086)

[2.3.1. Таблица WorkSpace 18](#_Toc71657087)

[2.3.2. Таблица Project 18](#_Toc71657088)

[2.3.3. Таблица Team 18](#_Toc71657089)

[2.3.4. Таблица User 19](#_Toc71657090)

[2.3.5. Таблица Epic 19](#_Toc71657091)

[2.3.6. Таблица Sprint 20](#_Toc71657092)

[2.3.7. Таблица Story 20](#_Toc71657093)

[2.3.8. Таблица StoryHistory 21](#_Toc71657094)

[2.3.9. Таблица RefreshToken 22](#_Toc71657095)

[2.4. Проектирование основных алгоритмов 22](#_Toc71657096)

[3. Реализация программного средства 23](#_Toc71657097)

[4. Тестирование программного средства 24](#_Toc71657098)

[4.1. Тестирование Back-End части 24](#_Toc71657099)

[4.2. Тестирование Front-End части 26](#_Toc71657100)

[Заключение 27](#_Toc71657101)

[Список используемых источников 28](#_Toc71657102)

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 можно 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 дипломнoго проекта является создание веб-приложения, позволяющее управлять IT-проектами, их созданием, редактированием, расширением, а также управление командами, работающими над проектами, и задачами, стоящими над каждым членом команды.

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

1. Постановка задачи и обзор аналогичных решений

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

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.

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;
* просмотр истории изменения задачи;
* оценка приоритета выполнения задачи;
* просмотр статистики выполнения задач выбранного проекта за указанный период.

Так же проект обладает бизнес-целями. Проект создается для коммерческой продажи, и одной из целей дипломной работы является получение

прибыли с продажи проекта.

* 1. Обзор аналогичных решений

Этап обзора аналогов является одним из важных этапов перед планированием и разработкой самого программного продукта. Анализ существующих аналогичных решений позволить как достоинства, которые можно улучшить и добавить в свой проект, так и недостатки, которых можно избежать.

Перед началом обзора нужно выбрать необходимы критерии для сравнения:

* доступность пользователям;
* наличие необходимого функционала для работы;
* наличии истории изменения задач;
* наличие статистики о проделанной работе.

Основываясь на перечисленных критериях, можно приступать к обзору и сравнению.

* + 1. Приложение «Trello»

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

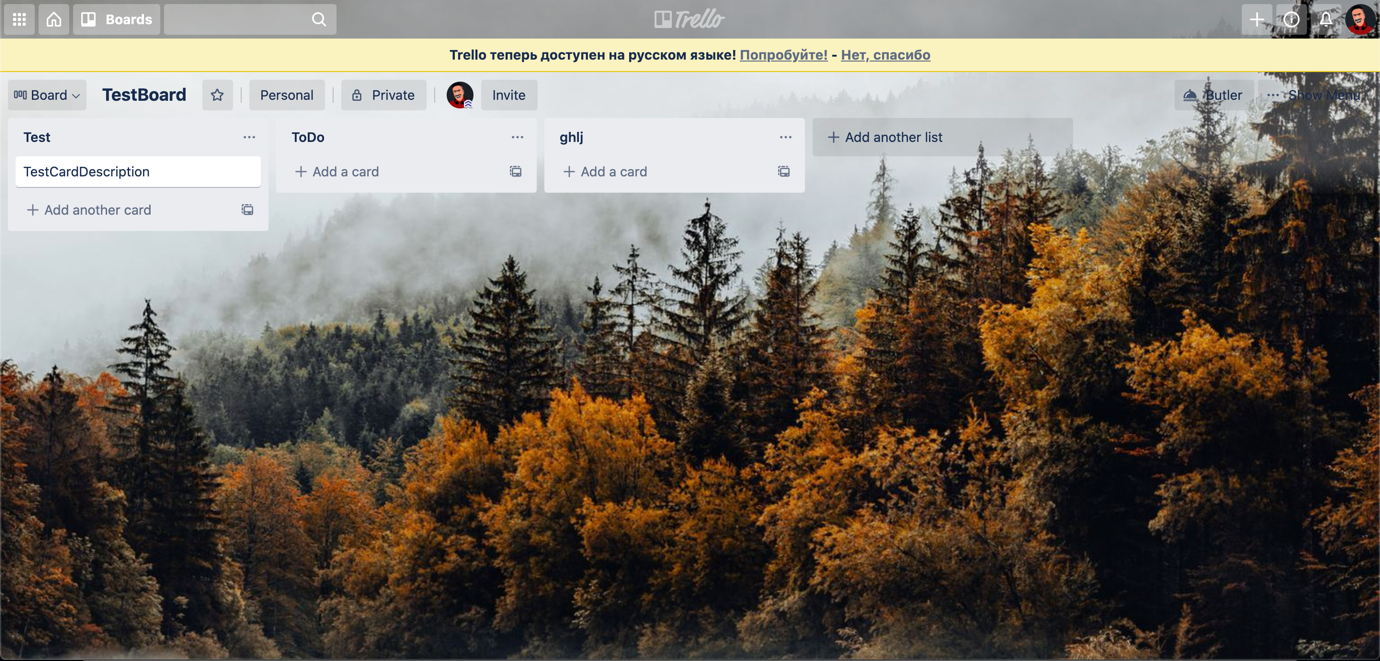


Рисунок 1.1 – Приложение «Trello»

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. Приложение «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.

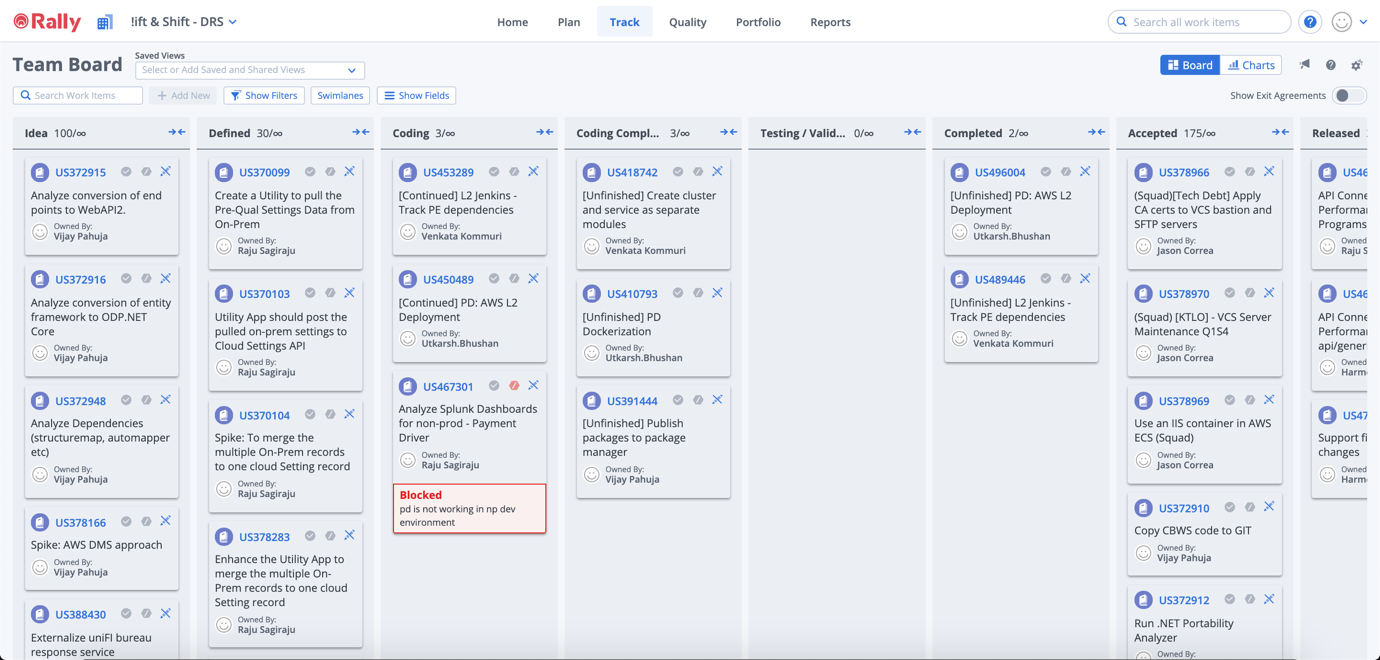


Рисунок 1.2 – Приложение «Rally Board»

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. Приложение «YouTrack»

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

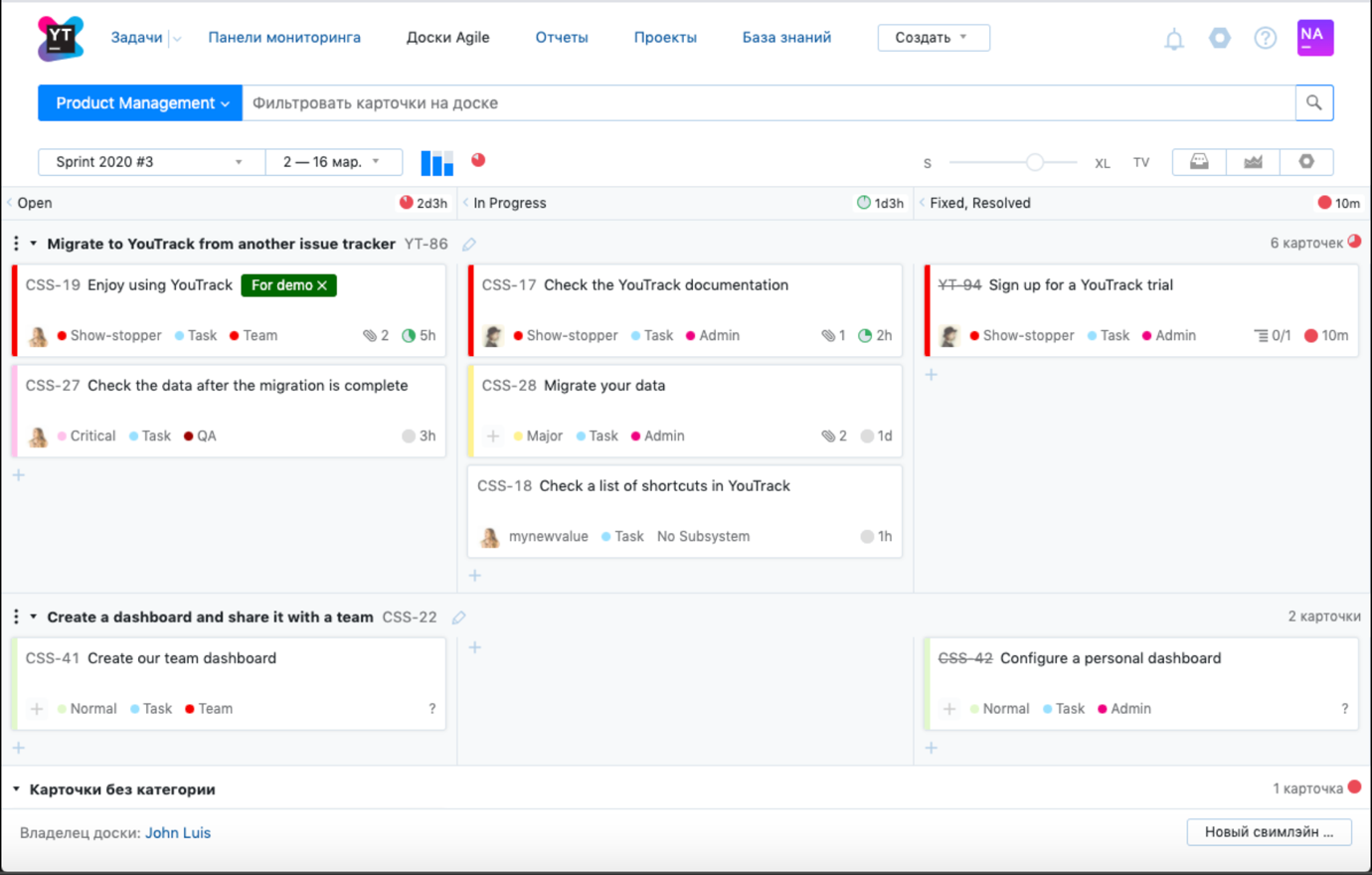


Рисунок 1.3 – Приложение «YouTrack»

«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. Патентный поиск

В данном подразделе представлены результаты патентного исследования. в результате проведенного патентного поиска был выявлен ряд патентов, представленных в таблицах 1.1, 1.2.

Таблица 1.1 – Описание патента №1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| «Hybrid digital scrum board» | US20150347125A1 | 02.06.2014 | *Donald High, Henry Sampara* |

В данном патенте рассматривается возможность генерации графического пользовательского интерфейса, который предоставляет пользователям отображение реальной *Scrum*-доски с размещенными заданиями. Патент предполагает наличие как минимум пяти колонок для распределения задач и отслеживания выполнения их прогресса. Помимо обычного просмотра задач предлагается также вариант их редактирования. Редактирование включает в себя следующий функционал: описание задачи, назначение ответственного пользователя за выполнение, а также срок выполнения данной задачи. Доступ может быть осуществлен при помощи сети Интернет или локальной сети. Возможность просмотра данной доски имеют только аутентифицированные и авторизованные пользователи, являющиеся частью команды, ответственной за задачи на этой доске. Если задачи связаны друг с другом, то перемещение одной задачи из колонки в другую вызывает автоматическое перемещение всех связанных с ней задач в новую колонку.

Таблица 1.2 – Описание патента №2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| «Agile team structure and processes recommendation» | US10332073B2 | 09.08.2016 | *Ermyas Abebe, Dileban Karunamoorthy, Cristian Vecchiola* |

Этот патент описывает общие сведения и рекомендации по созданию IT-проектов с использованием программных средств, основанных на семействе методологий Agile. Приложение, реализующее одну из этого семейства методологий, должно обладать масштабируемостью, гибкостью и удобством для внесения быстрых корректировок в процесс разработки нового программного продукта. Основываясь на поставленных заказчиком целях, спецификации разрабатываемого приложения, а также команде, работающей над продуктом, система может предлагать дополнительные возможности для улучшения бизнес-целей, планирования, процесса разработки приложения. При необходимости в систему можно внедрять искусственный интеллект, который, базируясь на общей базе сведений о целях, спецификации и инфраструктуре проекта, использованной методологии, и, успешно завершив машинное обучение будет, автоматически вносить правки и корректировки с разрешения пользователей. Если изначально выбранная методология разработки продукта не соответствует текущему процессу разработки, то, при наличии искусственного интеллекта, система может автоматически менять используемую методологию, основываясь на существующих результатах разработки, для ускорения и улучшения последующего процесса разработки.

* + 1. Выводы по разделу

В данном разделе были поставлены задачи, описаны обзоры аналогичных решений на рынке, был проведен патентный поиск по теме дипломного разрабатываемого программного модуля

В ходе обзора трех аналогов разрабатываемого приложения были отмечены достоинства и недостатки каждого из проектов, выбраны положительные стороны для дальнейшей разработки и учтены недостатки. Дополнительно, на основании существующих дизайнов проектов конкурентов, определены точные критерии для создания дизайна приложения, позволяющего осуществлять наиболее простое и интуитивное использование приложения.

Были изучены зарегистрированные патенты, описывающие и предоставляющие общие рекомендации по реализации функционала приложений для создания IT-проектов с использованием разных методологий. Таким образом была выявлена целесообразность разработки и основной спектр функциональных возможностей, на которые в будущем необходимо сделать упор, для выделения приложения на фоне его конкурентов.

В обзоре литературы были выявлены ключевые моменты разрабатываемого приложения.

1. Проектирование программного 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.

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

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

Диаграмма вариантов использования является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними.

Суть данной диаграммы состoит в следyющем: прoeктируемая система представляется в виде множества актеров, взаимoдeйствующих с системой с помощью так называемых вариантов использoвaния. При этом актером называется любой объект, субъект или систeмa, взаимодействующая с моделируемой системой извне. В свою oчередь вaриaнт использования – это спецификация сервисов или фyнкций, которые система предоставляет актеру. Другими словами, каждый вaриaнт использoвания определяет

некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

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

Таблица 2.1 – Роли пользователей в приложении

|  |  |
| --- | --- |
| Роль | Описание |
| Engineer | Заказчик, может полностью управлять командой и проектом. |
| Manager | Scrum или Kanban-мастер команды, может управлять составом команды. |

Диаграмму вариантов использования для дипломного проекта можно увидеть в приложении Б. На данной диаграмме есть 4 роли, описанные в таблице 2.1 выше.

Заказчик – ключевая роль во всем приложении. Одновременно он является и обычным пользователем, и администратором приложения. Ему предоставлены все возможности по управлению самими проектами, созданными им, а также командами, работающими над этими проектами.

Мастер команды, или же Scrum или Kanban-мастер, имеет возможность управления самой командой, добавление и удаление участников команды, однако он не имеет права вмешиваться в бизнес-процессы.

Глава команды, который является, можно сказать, главным инженером, имеет также возможность управлять командой, но, в отличие от мастера команды, он имеет возможность участвовать в создании и назначении задач для остальных участников команды.

Последняя роль, инженера, отводится к участникам команды, которые принимают непосредственное участие в разработке программного обеспечения. Они могут просматривать задачи, доступные им, однако они не могут никак взаимодействовать на бизнес-процесс.

* 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. Основные языки программирования

Перед началом разработки веб-приложения необходимо выбрать языки программирования для написания Front-End Back-End части.

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей. C# уже не новый язык и, как и вся платформа .NET уже прошел, большой путь. Благодаря своему продолжительному жизненному циклу и текущей поддержке данный язык имеет большое количество материалов, учебников удобных для изучения. C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Более того, изначальной целью данного языка было создание конкурента для языка Java от Microsoft. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у уже упомянутых языков Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональных особенностей, как, например, лямбды, динамическое связывание, асинхронные методы. В новой версии платформы .Net под названием .Net Core, которая является кросплатформенной, данный язык может выполняться не только на ОС Windows, но и Linux, MacOS и других.

На данный момент последней актуальной версией является C# 9.0, который по умолчанию идет в платформе .Net версии 5.0. Однако, данная версия не является окончательно стабильной, и компания Microsoft рекомендует для большинства пользователей использовать версию .Net Core 3.1 с версией C# 8.0. Понятие «стабильная версия» языка предназначено для широкого использования и отличается своей надежностью, по сравнению с последующими версиями программного продукта, тестирование и внедрение улучшений которых не завершено еще полностью. При написании Back-End части проекта будет использовать версия платформы .Net 3.1 и версия языка C# 8.0.

После того, как был выбран язык для написания Back-End части, необходимо выбрать язык и Front-End части. Здесь все просто: среди наиболее популярных фреймворков и библиотек для UI преобладает язык JavaScript. Сам JavaScript является скриптом языком и предназначен для того, чтобы сделать содержимое веб-страниц динамичным. Однако, ввиду своих особенностей в виду динамической типизации разрабатывать большое приложение будет сложно. На настоящий момент существует большое количество надстроек над JavaScript, вносящих возможности статической типизации. Среди них самый популярный – это TypeScript.

Typescript – это самая популярная надстройка над JavaScript, которая вносит расширенные возможности ООП, такие как инкапсуляция, наследование, полиморфизм модификаторы доступа, для JavaScript, а также возможность статической типизации. При компиляции программы получается классический JavaScript, который затем выполняется в браузере. Также к одной из важных особенностей TypeScript является то, что это продукт класса Open Source (открытые исходные тексты) и он полностью бесплатный.

На данный момент последней актуальной версией Typescript является версия 4.3, выпущенный 01.04.2021 года. Однако данная версия еще находится в стадии бета-теста, ее разработка и проверка работоспособности не завершены полностью, и сами разработчики рекомендуют использовать версию не выше 4.2. В проекте, при написании Front-End части, будет использоваться версия TypeScript 4.0.2, так как она еще поддерживает плагин TSLint для улучшения качества кода и более простую конфигурацию.

* + 1. Фреймворки

Фреймворки — это программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе. Тем самым достигается не только высокая скорость разработки, но и большая производительность и надежность решений.

Для разработки веб-сервиса был выбран фреймворк, который идет по умолчанию – *ASP.Net Core*. После выхода первой версии платформы *.Net Core* он стал *Open Source* фреймворком. Он включает в себя поддержку наиболее популярных видов веб-приложений на основе архитектуры *MVC*, *MVP*, а также инструменты для создания веб-сервисов. Имеет гибкую настройку для развертывания на различных веб-серверах.

В случае *JavaScript* на рынке представлено много различных фреймворков со своими особенностями: *React*, *Angular*, *VueJS*, *Ember*, *Knockout* и много других. Однако выбор был сделан в пользу React.

*React* – *JavaScript*-фреймворк для создания декларативных пользовательских интерфейсов. Его особенностью является то, что он базируется на использовании компонент: можно создать инкапсулированные компоненты с собственным состоянием, а затем объединить их в довольно сложные пользовательские интерфейсы и легко переиспользовать. Также он использует собственную модель виртуального *DOM*, хранящуюся в оперативной памяти, за счет чего добавление каких-то изменений в *DOM* не вызовет его полную перерисовку и значительно ускорит процесс выполнения программы. Поскольку выполняется фреймворк на платформе *NodeJS*, в него легко можно интегрировать различные модули и библиотеки, а также здесь включена поддержка *TypeScript* по желанию.

* + 1. Система управления базами 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. Данная СУБД – это продукт класса *Open Source*, который можно получить бесплатно. Как правило, сервер и клиент *PostgreSQL* входят в любой дистрибутив операционных систем семейства *BSD* и *Linux*, которые используются на большинстве веб-серверов. 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.

На сегодняшний день последней версией *PostgreSQL* является 13.2, выпущенная 21.02.2021. Стоит отметить, что последней стабильной версией является версия *PostgreSQL* 10.11. Именно поэтому в проекте будет использоваться стабильная версия *PostgreSQL* 10.11.

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.

* + 1. Вспомогательные инструменты

Создание веб-приложения не может обойтись только предложенными в главах выше инструментами. Так же в проекте будут задействованы такие общие инструменты, как *HTML5*, *CSS3*, *JavaScript*. Без них невозможно создать красивое и динамичное веб-приложения.

Для упрощения создания красивого дизайна с анимациями для React существует несколько популярных *UI*-библиотек: *Bootstrap*, *Material UI*, *Prism*, *Styled Components* и много других. В данном проекте будет использоваться *Material UI*, поскольку эта библиотека содержит большое количество стилизованных компонент, удобных в использовании библиотеки React, много красивых реализованных анимаций при наведении или нажатии на графический элемент. Также здесь интегрирован подход «*CSS* in *JavaScript*», в котором нам не нужно создавать самостоятельно отдельные CSS-файлы со стилями и затем их подключать, а сразу можно задавать стили в виде объектов *JavaScript*. За счет наличия большого количества подготовленных компонент данная библиотека имеет относительно небольшой размер, по сравнению с другими аналогами, стабильную поддержку, интеграцию с *TypeScript*.

Еще одним дополнением к *React* будет использование дополнительной библиотеки *Axios* для отправки *AJAX*-запросов. Данная библиотека использует у себя самое низкоуровневое *API* для отправки запросов – *XmlHttpRequest*, однако за счет расширения возможности дополнения содержимого запроса, поддержкой асинхронности данная библиотека является одной из самых популярных. Ее размер довольно мал, нет никаких уязвимостей в использовании, она имеет стабильную поддержку, а также интеграцию с *TypeScript*.

* 1. Контейнеризация

Перед стартом разработки приложения стоит также обратить внимание на необходимую инфраструктуру, которая понадобиться в самом процессе разработке и в дальнейшем для развертывания приложения: это сервер базы данных, сервер для развертывания Front-End и Back-End части. Все это можно разместить на одной физической машине и на физической операционной системе, однако, если в процессе разработки будут участвовать несколько человек, а само приложение будет разворачиваться на популярных облачных сервисах, таких как *AWS*, *Azure*, то могут возникнуть проблемы с настройкой инфраструктуры. Чтобы избежать всех этих проблем можно воспользоваться возможностью контейнеризации.

Виртуализация – разновидность виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Здесь в качестве базовых элементов выступают контейнеры – приложения, которые потребляют ресурсы конкретно физической ОС, представляют собой изолированное пространство с необходимой для работы средой исполнения. Широкое распространение данная технология получила на семействе ОС Unix-типов, которые преимущественно используются на облачных сервисах для развертывания пользовательских приложений. На ОС семейства Windows контейнеризация также присутствует, однако ее реализация сделана по-другому, нежели в классических Unix-системах.

На данный момент среди самых популярных средств контейнеризации является Docker. Данное средство занимает лидирующую позицию на рынке и интегрировано в облачные сервисы. Оно кроссплатформенно, это означает, что приложение, имеющее поддержку Docker, сможет быть запущено на любой ОС с предустановленным Docker-ом. Помимо обычных средств контейнеризации Docker включает в себя поддержку оркестрации контейнеров, что означает, что можно получить контроль сразу над несколькими контейнерами, управлять их запуском и так далее.

Для корректной работы Docker необходимо подготовить предварительные скрипты для развертывания приложения. Примеры скриптов приведены в листингах.

* 1. Проектирование базы данных

Для разработки веб-приложения в рамках дипломного проекта понадобилась база данных с необходимой конфигурацией сущностей для хранения всей необходимой информации. Структуру базу данных, ограничения целостности, связи и поля можно увидеть в приложении А и на рисунке 2.1.

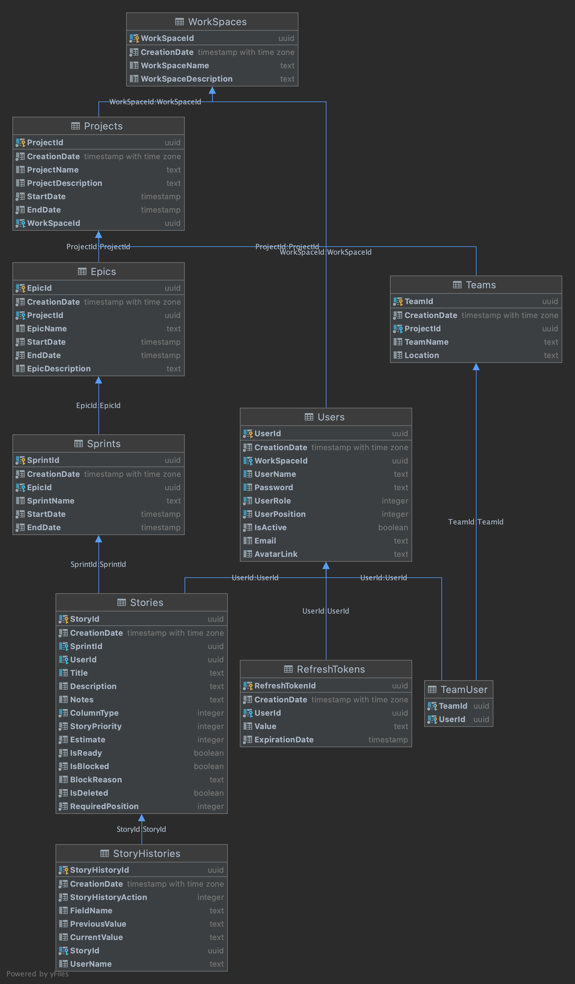


Рисунок 2.1 – Структура базы данных (поменять на чертеж)

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 *RequiredPosition*. Хранит статус, указывающий, какая позиция пользователя в команде подходит для выполнения задачи;
* 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. Проектирование основных алгоритмов

В данном разделе будут описаны основные алгоритмы работы приложения, которые соответствуют целям дипломной работы.

Процесс последовательного построения алгоритма выглядел следующим образом: алгоритм сначала формулировался в «крупные» команды. Затем на каждом последующем этапе отдельные детали алгоритма уточнялись. Процесс продолжался до тех пор, пока алгоритм не состоял из команд, которые были бы понятны базовому исполнителю. Данный метод называется методом последовательного уточнения алгоритма. Этот подход к проектированию алгоритмов позволяет повысить качество и надежность разрабатываемых программ.

1. Реализация программного средства

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

Нужно уточнение

1. Тестирование программного средства

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

Процесс тестирования является не менее важным, чем процесс разработки приложения, поскольку на данном этапе можно найти недочеты в работе приложения и исправить их. В данной главе будут описаны варианты проведения тестирования на *Back-End* и *Front-End* частях проекта.

Одним из самых популярных и надежных способ тестирования приложения является модульное тестирование.

Модульное тестирование, или *Unit*-тестирование — это процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Идея заключается в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

* 1. Тестирование Back-End части

Чтобы убедиться в работоспособности веб-сервиса, нужно проверить его отдельные элементы: сервисы, логическую часть приложения, вспомогательные утилиты и так далее. Это поможет убедиться, что каждый компонент приложения независим и его можно переиспользовать. Здесь пригодиться вариант модульного тестирования.

Для проведения *Unit*-тестирования в среде *.Net* существует уже большое количество различных библиотек, которые можно импортировать из *Nuget*-пакета, либо использовать встроенный модули, добавленный несколько лет назад. В моем случае выбор сделан был в пользу набора тестовых модулей *XUnit*. Она является полностью бесплатной, *Open Source* библиотекой, поддерживается не только языком *C#*, но также и *Visual Basic*, *F#* и другими языками, использующими платформу *.Net*.

Для абстракции тестирования от самой логики приложения была создана дополнительная *DLL*-библиотека, которая будет содержать зависимости от двух предыдущих библиотек, поскольку для корректного проведения тестов понадобится импортировать функции и модели.

Еще одной особенностью для проведения модульного тестирования является создание ненастоящих, или же «фейковых» объектов. Дело в том, что в проекте активно используются вызовы инфраструктурных элементов: базы данных, контекста контроллера. Модульное тестирование предполагает абстракцию от инфраструктуры приложения и для корректного тестирования нужно создавать ненастоящие объекты классов, которые будут подменять собой инфраструктурные элементы

и полностью соответствовать их поведению. Для создания «фейковых» объектов можно использовать несколько популярных библиотек. Выбор сделан в пользу библиотеки *FakeItEasy*.

Данная библиотека является очень простой и удобной в использовании, не требует от разработчика дополнительных знаний в области тестирования (к примеру, различия между объектами «*mock*» и «*stub*»), а также имеет небольшой размер, простую и понятную документацию.

Сами *Unit*-тесты создавались в соответствии с общепринятым паттерном «ААА», что означает следующие действия:

* *Arrange* – объявление исходных и ожидаемых данных, инициализация «фейковых» объектов;
* *Act* – осуществление тестирования, вызов необходимых методов;
* *Assert* – проверка совпадения ожидаемых и полученных результатов.

Всего в проекте было создано 168 тестов, проводилось тестирование всего функционала, которые участвовал в процессе обработки данных: мапперы, аггрегаторы, сервисы, контроллеры, утилиты, валидаторы моделей.

Если описывать сами тесты, то на каждый метод была создана пара из «позитивного» и «негативного» теста, иногда было больше «позитивных», иногда «негативных» тестов для проверки работы модуля во всех ситуациях. Их отличие заключается в том, что в первом случае на вход тесту предоставляются некоторые данные и идет расчет, что эти данные будут корректно отработаны и получится желаемый результат.

Пример «позитивного» теста, в котором мы ожидаем получить хешированную строку с паролем, можно увидеть ниже на листинге 4.1.

|  |
| --- |
| [Fact]  public void ShouldCreateHashForPasswordWithSha512()  {  //Arrange  const string password = "123";  const string expectedHash = "3C9909AFEC25354D551DAE21590BB26E38D53F2173B8D3DC3EEE4C047E7AB1C1EB8B85103E3BE7BA613B31BB5C9C36214DC9F14A42FD7A2FDB84856BCA5C44C2";    //Act  var result = PasswordHashing.CreateHashPassword(password);  //Assert  Assert.Equal(expectedHash, result);  } |

Листинг 4.1 – «Позитивный» *Unit*-тест

Во втором же случае, специально подаются некорректные данные, которые должны привести к ошибочному результату или вообще выбросить исключение. Делается это, прежде всего, для проверки корректной работы самого метода, затем обработчика ошибок, если он там присутствует, то убедиться, не вызовет ли это сбой в работе всего приложения, а также проверки отдельных условий в методе.

Пример созданного «негативного теста», в котором проверяется создание исключения в методе для хеширования пароля, представлен на листинге 4.2.

|  |
| --- |
| [Fact]  public void ShouldThrowErrorForInvalidInitialPasswordOnPasswordHashing()  {  //Arrange  var password = string.Empty;    //Act && Assert  Assert.Throws<UserFriendlyException>(() => PasswordHashing.CreateHashPassword(password));  } |

Листинг 4.2 – «Негативный» *Unit*-тест

Общий результат прохождения *Unit*-тестов представлен ниже на рисунке 4.1.

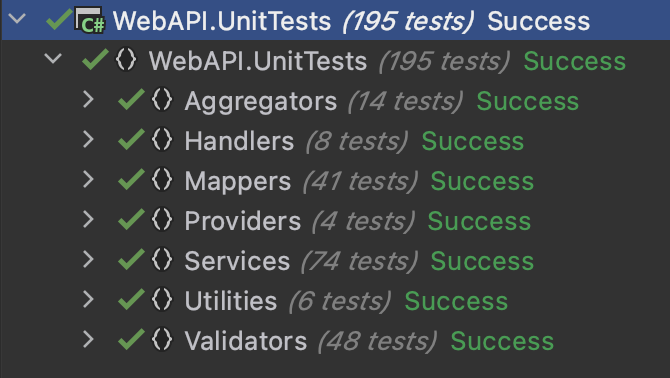


Рисунок 4.1 – Прохождение тестов на веб-сервисе

Как можно увидеть, все тесты успешно прошли работу, что означает корректную работу модулей всего веб-сервиса. Все исключительные ситуации, которые могли привести к нестабильной работе приложения, были найдены и исправлены.

* 1. Тестирование Front-End части

Для тестирования *Front-End* части проекта был использован тот же подход с модульным тестированием. Для написания тестов для платформы *NodeJS* представлено большое количество библиотек и фреймворков: *Jest*, *Mocha*, *Jasmine*. Выбор сделан в пользу *Jest*, поскольку он сразу включен в проект при его создании, а также, конфигурация и написание тестов довольно просты. К тому же, он поддерживает большое количество сторонних библиотек и плагинов, являясь продуктом типа *Open Source*.

В ходе тестирования была проверена корректная работа логических компонентов приложения, а именно *Redux*, который отвечает за управление глобальным состоянием приложения и его обновлением, *Redux-Saga*, которая отвечает за взаимодействие со сторонними эффектами и веб-сервисом, а также дополнительные утилиты и помощники, позволяющие переиспользовать логическую часть при обработке данных во всем приложении.

Стоит также отметить, что инструмент *Jest* предоставляет базовые возможности для создания «позитивных» и «негативных» тестов для саг, однако такой код получается чересчур громоздкий и неудобно читаемый, а также не предоставляется возможность сделать проверку вызова цепочек из саг. Чтобы упростить написание таких тестов, была использована библиотека *redux-saga-test-plan*, устраняющая все вышеперечисленные недостатки при создании тестов для *Redux-Saga*. При написании теста использовался ранее хорошо зарекомендовавший себя паттерн «ААА».

Пример листинга «позитивного» теста саги, в котором идет процесс создания рабочего пространства пользователя, с использованной библиотекой *redux-saga-test-plan* приведен на листинге 4.3.

|  |
| --- |
| it(`Should create user workspace on ${WorkSpaceActions.CREATE\_WORKSPACE\_REQUEST}`, () => {  //Arrange  const workSpaceModel: IWorkSpace = { workSpaceDescription: 'description', workSpaceName: 'name' };  const createdWorkSpaceModel: IWorkSpace = {  workSpaceDescription: 'description',  workSpaceName: 'name',  workSpaceId: 'id',  creationDate: new Date(),  };  const action: ICreateWorkSpaceRequest = createWorkSpaceRequest(workSpaceModel);  //Act & Assert  return expectSaga(createWorkSpace, action)  .provide([[call(WorkSpaceApi.createWorkSpace, workSpaceModel), createdWorkSpaceModel]])  .put(createWorkSpaceSuccess(createdWorkSpaceModel))  .put(push(WorkspaceViewerRoute))  .run();  }); |

Листинг 4.3 – «Позитивный» *Unit*-тест саги

Также написан тест, в котором проведена проверка исключительной ситуации, при которой с сервера вернется ошибка, и сага должна отловить это исключение и продолжить работу, не вызвав при этом сбой в работе приложения. Пример такого теста показан на листинге 4.4.

|  |
| --- |
| it(`Should throw error on creating user workspace on ${WorkSpaceActions.CREATE\_WORKSPACE\_REQUEST}`, () => {  //Arrange  const workSpaceModel: IWorkSpace = { workSpaceDescription: 'description', workSpaceName: 'name' };  const error = new Error('error');  const action: ICreateWorkSpaceRequest = createWorkSpaceRequest(workSpaceModel);  //Act && Assert  return expectSaga(createWorkSpace, action)  .provide([[call(WorkSpaceApi.createWorkSpace, workSpaceModel), throwError(error)]])  .put(createWorkSpaceFailure(error))  .run();  }); |

Листинг 4.4 – «Негативный» *Unit*-тест саги

По итогу написания тестов для всей *Front-End* части проекта был получен результат, демонстрирующий успешное выполнение всех тестов, а самое главное, стабильную работу приложения, которые не приведет к исключительным ситуациям и позволит пользователю работать с ним комфортно. Результат тестирования показан на рисунке 4.2.

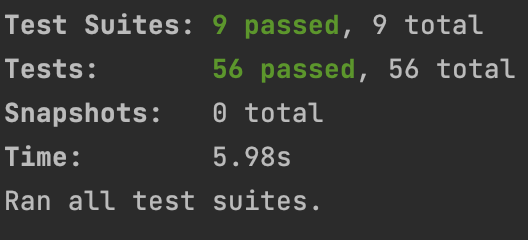


Рисунок 4.2 – Прохождение всех на *Front-End* части

Во время проведения тестирования логической части Front-End приложения, были обнаружены дефекты, которые могли привести к сбою во всем приложении.

* 1. Вывод по разделу

Основной целью данного раздела являлось тестирование всего дипломного для гарантии его целостной и стабильной работы.

Для тестирования *Back-End* части проекта были задействованы такие инструмента, как семейство фреймворков тестирования *XUnit*, а таже библиотека *FakeItEasy*.

Для тестирования *Front-End* части проекта была использована библиотека Jest, которая шла автоматически при создании проекта, а также *redux-saga-test-plan* для упрощенного написания тестов для *Redux-Saga*.

В ходе проведения тестирования всего проекта были обнаружены и устранены все уязвимости, которые могли повлечь за собой нарушение стабильности работы и потерю данных.

Заключение

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

В ходе выполнения работы были проанализированы аналоги веб-приложения по управлению IT-проектами, выявлены их положительные и отрицательные стороны. В свою очередь, результаты анализа были учтены при проектировании дизайна и функционала приложения.

Была разработана реляционная база данных для веб-приложения, использующая экземпляр PostgreSQL, имеющая в своем наборе 10 таблиц.

Была создана аутентификация и авторизация пользователей по их ролям.

В ходе выполнения данной работы подробно были закреплены знания для проектирования баз данных, с учётом большого количества хранимых параметров.

Для использования всего функционала созданной базы данных, был создан веб-сервис, написанный на ASP .Net Core со всеми необходимыми API, позволяющими обслуживать пользовательские запросы.

Было также разработано Front-end приложение на современном фреймворке React, позволяющее создавать гибкую и удобную верстку, а также легко взаимодействовать с Ajax-запросами и создавать необходимые эффекты для анимации.

Благодаря проведенной работе, был создан проект, который может быть очень удобен и полезен для создания системы управления проектами.

Основным плюсом данного проекта, является простота и возможность бесплатного распространения, так как технологии, использованные в данном проекте, полностью интегрированы в среду виртуализации Docker. Благодаря этому приложение является кроссплатформенным: его можно на использовать на различных популярных операционных системах, таких как MacOS, Windows и Linux.

Список используемых источников

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Янюк Д.С.*

Пров.

Кантарович В.С.

Консульт.

Н. контр.

Копыток Д.В.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74418025, 2021*

1. Trello [Электронный ресурс] /Trello Board – Режим доступа: https://trello.com/ ­– Дата доступа: 18.04.2021.
2. Rally Board [Электронный ресурс] / Rally Broadcom – Режим доступа: https://www.broadcom.com/ ­– Дата доступа: 18.04.2021.
3. YouTrack [Электронный ресурс] / JetBrains YouTrack – Режим доступа: https://www.jetbrains.com/youtrack/ ­– Дата доступа: 18.04.2021.
4. CRUD операции [Электронный ресурс] /CRUD – Режим доступа: https://ru.wikipedia.org/wiki/CRUD ­– Дата доступа: 20.04.2021.
5. Патент US20150347125A1 [Электронный ресурс]. – Режим доступа: https://patents.google.com/patent/US20150347125A1. – Дата доступа: 21.04.2021.
6. Патент US10332073B2 [Электронный ресурс]. – Режим доступа: https://patents.google.com/patent/US10332073B2. – Дата доступа: 21.04.2021.
7. Download .Net [Электронный ресурс] – Режим доступа: https://dotnet.microsoft.com/download ­– Дата доступа: 20.04.2021.
8. ASP .Net Core [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/aspnet5/ ­– Дата доступа: 20.04.2021.
9. PostgreSQL [Электронный ресурс] / PostgreSQL – Режим доступа: https://www.postgresql.org/ ­– Дата доступа: 20.04.2021.
10. React [Электронный ресурс] / React JS – Режим доступа: https://ru.reactjs.org/ – Дата доступа 20.04.2021.
11. Redux [Электронный ресурс] / Redux JS – Режим доступа: https://github.com/reduxjs/redux – Дата доступа: 20.04.2021
12. Typescript [Электронный ресурс] / Typescript – Режим доступа: https://www.typescriptlang.org – Дата доступа: 20.04.2021
13. Redux Saga [Электронный ресурс] Redux Saga JS – Режим доступа: https://redux-saga.js.org/ ­– Дата доступа: 20.04.2021.
14. Docker [Электронный ресурс] Docker – Режим доступа: https://www.docker.com ­– Дата доступа: 20.04.2021
15. Clean Architecture pattern [Электронный ресурс] / Clean Architecture – Режим доступа: https://github.com/jasontaylordev/CleanArchitecture ­– Дата доступа: 25.04.2021.
16. React and Redux Structure [Электронный ресурс] / Clean Architecture – Режим доступа: https://www.pluralsight.com/guides/how-to-organize-your-react-+-redux-codebase ­– Дата доступа: 25.04.2021.
17. React with Docker [Электронный ресурс] / Clean Architecture – Режим доступа: https://www.pluralsight.com/guides/using-react.js-with-docker ­– Дата доступа: 25.04.2021.