

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

<i>Факультет</i>	<i>Кафедра</i>
вычислительной техники	МОиПЭВМ
<hr/>	<hr/>
(наименование)	(наименование)
Направление подготовки	09.03.04 «Программная инженерия»
	<hr/>
	(код и наименование)

БАКАЛАВРСКАЯ РАБОТА

на тему

«РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ
РАЗРАБОТКОЙ ПРОГРАММНЫХ ПРОЕКТОВ ПО МЕТОДОЛОГИИ SCRUM»

Студент	<hr/>	Алпатов Даниил Михайлович
	(подпись, дата)	(фио полностью)
Руководитель	<hr/>	Кольчугина Е.А.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	<hr/>	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	<hr/>	Макарычев П.П.
	(подпись, дата)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	<hr/>	Попова Н.А.
	(подпись, дата)	(фамилия, инициалы)

Пенза, 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«__» _____ 2020 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Алпатов Даниил Михайлович гр.16ВП1 факультета ВТ

2. Тема работы «Разработка веб-приложения для организации управления разработкой программных проектов по методологии Scrum»

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы Кольчугина Е.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение:

- создание и отслеживание статуса задач для разработки
- предоставление инструментов для планирования спринтов

Цель создания веб-приложения:

предоставление пользователям вспомогательного инструмента для применения методологии Scrum

Основные функции веб-приложения:

- а) создание и редактирование существующих задач
- б) создание спринтов и наполнение их задачами
- в) система личных сообщений

Структуры данных: пользователь, задача, спринт, сообщение, изображение, страница.

Технология разработки: ООП, MVC

Языки программирования: TypeScript, SQL, C#, HTML5, CCS3

Среда исполнения: веб-сервер IIS, веб-браузер Google Chrome, СУБД MySQL

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области
2) Анализ функциональных требований
3) Планирование разработки и оценка бюджета
4) Проектирование веб-приложения
5) Реализация
6) Контроль качества
- тестирование
- метрики кода
7) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	15	23.11.2019 - 17.12.2019	
2	Анализ функциональных требований	10	21.12.2019 - 24.01.2020	
3	Планирование разработки и оценка бюджета	10	07.02.2020 - 25.02.2020	
4	Проектирование веб-приложения	20	01.03.2020 - 30.03.2020	
5	Реализация	30	02.04.2020 - 14.05.2020	
6	Контроль качества	5	16.05.2020 - 23.05.2020	
7	Оформление пояснительной записки	10	25.05.2020 - 15.06.2020	

Дата выдачи задания

"__" _____ 20__

Руководитель бакалаврской работы

(подпись, дата)

Задание к исполнению принял студент

(подпись, дата)

Работу к защите допустить
Декан факультета

(подпись, дата)

Реферат

Пояснительная записка содержит 105 листов, 22 рисунка, 30 таблиц, 27 источников, 13 приложений.

АНАЛИЗ, ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА, ТЕСТИРОВАНИЕ, ПЛАНИРОВАНИЕ, ОЦЕНКА БЮДЖЕТА, ВЕБ-ПРИЛОЖЕНИЕ, СИСТЕМА УПРАВЛЕНИЯ, SCRUM, МЕТОДОЛОГИИ РАЗРАБОТКИ.

Объект исследования – автоматизированные информационные системы.

Цель работы – проектирование и разработка веб-приложения система управления разработкой программных проектов по методологии SCRUM.

Технология разработки: ООП, MVC, IIS, C#, ASP.NET Core, Entity Framework Core, MS Test, TypeScript, Angular.

В результате выполнения работы было спроектировано и создано веб-приложение система управления разработкой программных проектов по методологии SCRUM. Отличительной особенностью системы является специализация на создании задач и планировании спринтов, а также на взаимодействии пользователей.

Основные технико-эксплуатационные характеристики:

- универсальность;
- широкие функциональные возможности;
- программная совместимость с многими операционными системами;
- возможность быстрой модификации.

Данный программный продукт предназначен для использования в качестве вспомогательного инструмента при следовании методологии SCRUM в процессе разработки программных проектов. Веб-приложение может применяться командами разработчиков любого уровня.

					ПГУ 09.03.04–8ВР131.03 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Разработка веб-приложения для организации управления разработкой программных проектов по методологии SCRUM» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Алпатов Д.М.						
Провер.		Кольчугина Е.А.					4	105
Н. Контр.		Такташкин Д.В.				Группа 16ВП1		
Утверд.		Макарычев П.П.						

Содержание

	Стр.
Введение.....	7
1 Исследование предметной области.....	9
1.1 Описание предметной области	9
1.2 Обзор научных исследований о методологии Scrum	10
1.3 Обзор сервисов и систем для применения Scrum.....	11
1.4 Сравнительный анализ аналогов	15
1.5 Анализ вариантов использования.....	19
1.6 Структура приложения	25
1.7 Техническое задание.....	26
2 Планирование разработки и оценка бюджета.....	31
2.1 Календарное и ресурсное планирование	31
2.2 Расчет бюджета	32
2.3 Выбор программно-аппаратных средств	37
3 Проектирование системы управления	40
3.1 Проектирование пользовательского интерфейса.....	40
3.2 Организация процесса обработки информации.....	42
3.3 Структуры хранения данных	43
4 Реализация системы управления	49
4.1 Кодирование системы управления.....	49
4.2 Развёртывание веб-приложения	55
5 Анализ качества программного продукта	57
5.1 Тестирование	57
5.2 Метрики кода.....	63
Заключение	66
Список использованных источников	68
Приложение А. Листинг кода файла Program.cs.....	70
Приложение Б. Листинг кода файла Startup.cs	72
Приложение В. Листинг кода файла Extensions.cs	75

Приложение Г. Листинг кода файла AuthRepository.cs	77
Приложение Д. Листинг файла UsersController.cs.....	80
Приложение Е. Листинг кода файла user.service.ts.....	83
Приложение Ж. Листинг кода файла messages.component.ts	87
Приложение И. Листинг кода маршрутизатора routes.ts	90
Приложение К. Листинг кода главного модуля app.module.ts	92
Приложение Л. Листинг кода юнит-теста для модуля работы с задачами.....	96
Приложение М. Листинг кода сервиса аутентификации auth.service.ts	100
Приложение Н. Распределение трудовых ресурсов	102
Приложение П. Диаграмма Ганта.....	105

Введение

Разработка программного обеспечения на сегодняшний день является важной сферой деятельности человека. Поэтому для повышения эффективности этого вида деятельности были разработаны специальные методологии. С течением времени разрабатывались их новые виды, а уже существующие претерпевали изменения. Сегодня самыми популярными являются так называемые «гибкие методологии» («agile»). Одна из них – SCRUM – настолько популярна среди разработчиков программного обеспечения, что зачастую SCRUM и Agile используют как синонимы.

SCRUM – это фреймворк, в рамках которого люди могут решать сложные адаптивные проблемы, одновременно производя и поставляя продукты максимально возможной ценности [1].

SCRUM является простой в понимании методологией, так как содержит небольшое количество сущностей и малое количество обязательных предписаний, из-за чего и обрела свою популярность.

Согласно теории методологии SCRUM не обязательно пользоваться какими-либо инструментами. Однако в мире информационных технологий команды, работающие над проектом, часто разделены большими расстояниями, что усложняет коммуникацию. Поэтому на практике использование специализированных инструментов значительно повышает эффективность методологии SCRUM, а значит актуальность таких инструментов очевидна.

Существующие инструменты зачастую перегружены ненужными функциями и предоставляют бесплатную версию только для небольшого количества пользователей. Одна из главных идей SCRUM – облегчать работу разработчиков, а не нагружать их изучением и использованием дополнительных инструментов, не влияющих напрямую на качество производимого программного продукта. Для создания легковесного и эффективного вспомогательного инструмента для SCRUM необходимо определить и реализовать функции, предполагаемые методологией.

Объект исследования – автоматизированные информационные системы.

Предмет – управления разработкой программных проектов по методологии SCRUM.

Целью работы является проектирование и разработка веб-приложения для управления разработкой программных проектов по методологии SCRUM.

Для достижения поставленной цели были определены следующие задачи:

- подобрать и изучить литературу по проектированию и разработке веб-приложений;
- проанализировать степень разработанности проблемы отечественными и зарубежными авторами;
- проанализировать предметную область;
- провести анализ существующих систем управления по интегральным и частным критериальным оценкам;
- определить структуру и содержание веб-приложения для управления;
- осуществить проектирование пользовательского интерфейса, программных средств и структур данных разрабатываемой системы;
- провести планирование и расчёт бюджета разработки;
- реализовать веб-приложение для управления разработкой программных проектов по методологии SCRUM;
- провести тестирование разработанного приложения.

1 Исследование предметной области

1.1 Описание предметной области

Scrum является простой в понимании методологией, так как содержит небольшое количество сущностей и малое количество обязательных предписаний, из-за чего и обрела свою популярность.

Методология Scrum включает в себя пять различных событий. Ниже представлены их краткие описания.

Спринт – период времени, за который создается новая рабочая итерация продукта. В зависимости от решения команды определяется в диапазоне от одной до четырех недель. Является контейнером для остальных четырех событий, которые происходят во время спринта.

Планирование спринта – событие, во время которого команда определяет, над чем и как будет работать в течение спринта.

Ежедневный Scrum – ежедневная встреча всех участников команды для обсуждения текущего прогресса и корректировки планов.

Обзор спринта – событие, которым завершается спринт. Команда обсуждает прогресс, которого смогла добиться за прошедший спринт. К обсуждению могут присоединиться акционеры и владельцы бизнеса.

Ретроспектива спринта – событие, во время которого команда обсуждает, как она может стать эффективнее. Именно во время ретроспективы могут вноситься изменения в организационный процесс.

Методология Scrum включает в себя три артефакта. Ниже представлены краткие описания каждого из них.

Бэклог продукта – единственный список задач для команды. Он может включать требования, улучшения, новые возможности, дефекты и т.д. Владелец продукта – ответственный за его содержание, однако каждый член команды может вносить в него изменения.

Бэклог спринта – подмножество бэклога продукта, с задачами, которые должны быть выполнены в течение спринта, а также план команды по их

реализации. Создается командой разработки так, чтобы в результате выполнения всех задач получалась рабочая версия продукта.

Инкремент – сам продукт. Текущая версия продукта, имеющая ценность для бизнеса, и которая может использоваться. Владелец продукта решает, готов ли инкремент для выхода на рынок. Однако инкремент должен всегда быть в таком состоянии, чтобы быть готовым выйти на рынок. То есть должны отсутствовать критические недоработки и должен быть реализован весь необходимый для корректной работы функционал.

1.2 Обзор научных исследований о методологии Scrum

Проведем анализ литературы о методологии разработки Scrum.

Впервые термин появился в книге Нонаки Икуджиро и Такеучи Хиротаки «Компания – создатель знания», в которой авторы сравнивают команды разработки с конструкцией схватки (от англ. scrum) в регби, назначаемой судьей при остановке игры или при нарушении правил [2]. В книге рассматриваются истории успеха проектов, разработкой которой занимались небольшие команды без жесткой специализации.

Позже, в 1993 году американский программист Джеф Сазерленд применил этот подход, когда разрабатывал методологию для компании «Easel». В книге «Scrum – революционный метод управления проектами» он детально рассматривает применение этого подхода [3]. Именно тогда он и назвал его официально «Scrum». Спустя два года разработчик и консультант по разработке программного обеспечения Кен Швабер формализовал этот процесс применительно ко всей индустрии в целом и описал способы его применения в книге «Scrum – Гибкое управление продуктом и бизнесом» [4].

В 1995 Швабер выступил на конференции «Объектно-ориентированные системы, языки и приложения для программирования», где указал основу методологии и ее характеристики.

Проанализировав наиболее важные работы, можно сделать вывод, что тема применения методологии освещена в достаточном объеме. Однако

практически отсутствуют работы, рассматривающие применение конкретных вспомогательных инструментов и их сравнительный анализ. А также отсутствуют работы, описывающие создание этих инструментов.

1.3 Обзор сервисов и систем для применения Scrum

Для анализа предметной области необходимо определить основные аналоги.

На рынке существует несколько решений, позволяющих разработчикам внедрять методологию Scrum в свой рабочий процесс. Однако многие из них направлены на профессиональных разработчиков крупных продуктов.

Для анализа аналогов среди множества продуктов можно выделить некоторые из самых популярных решений:

- Trello;
- LeanKit;
- Jira;
- nTask;
- Asana.

Рассмотрим подробно их особенности ниже.

Начнем с сервиса для создания канбан-досок Trello [5].

Trello был разработан компанией Atlassian как замена бумажных канбан-досок [6, 7]. И хотя этот сервис изначально не предполагался для применения методологии Scrum, некоторые разработчики используют его именно для этих целей. Пример такого использования представлен на рисунке 1. Гибкая система надстроек позволяет добавить необходимый функционал, но общий интерфейс сервиса не был рассчитан для такого использования, поэтому Trello является не очень удобным инструментом для этих целей.

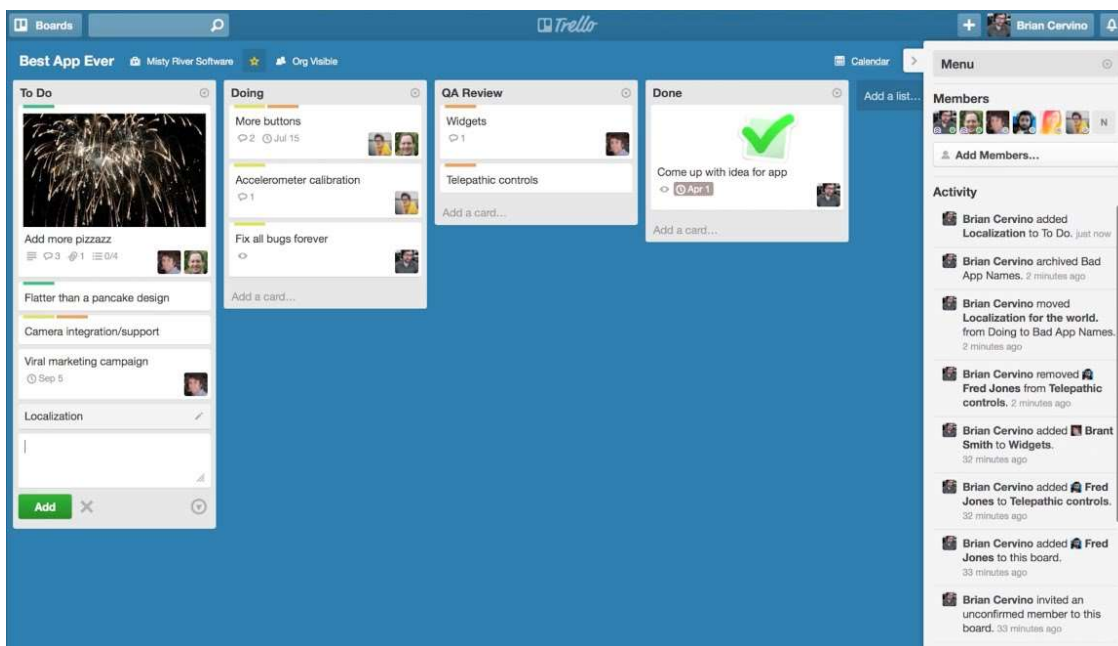


Рисунок 1 – Интерфейс Trello

LeanKit – еще одна альтернатива бумажных канбан-досок, однако не имеет такой гибкой системы для добавления надстроек, что делает этот инструмент еще менее удобным [8]. Рабочая область представлена на рисунке 2. И даже несмотря на это, применяется в рамках Scrum, хоть и с ограничениями и даже изменениями некоторых концепций.

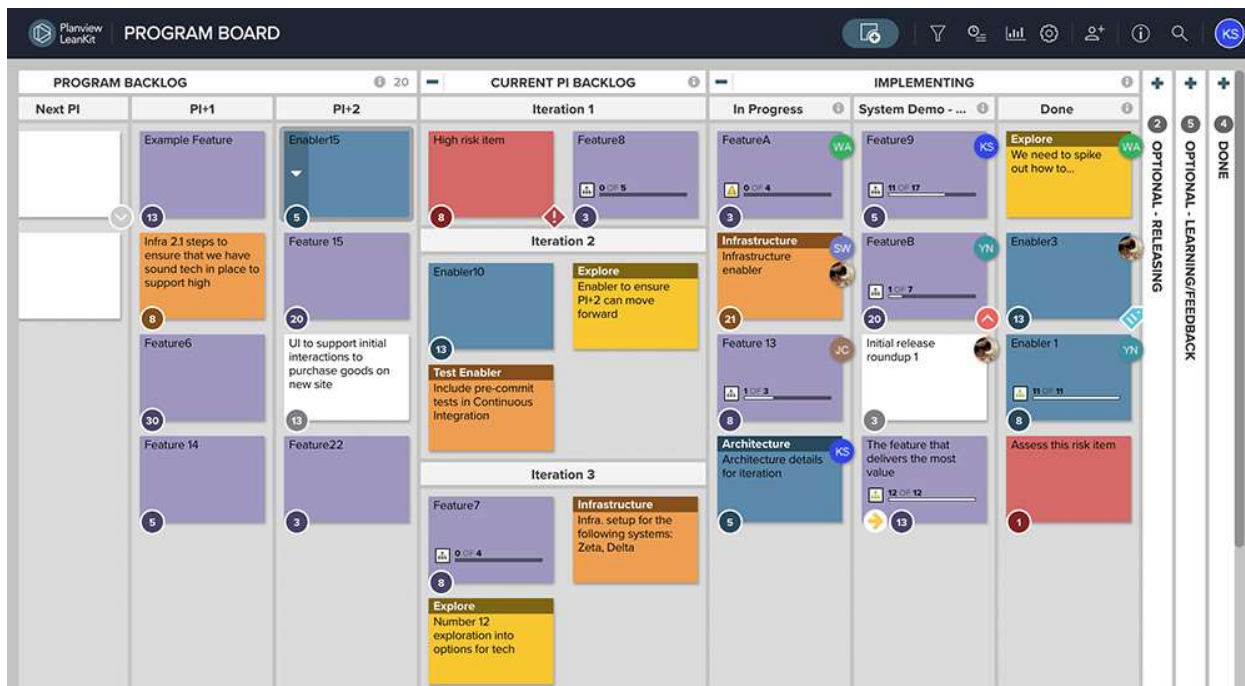


Рисунок 2 – Интерфейс LeanKit

Jira – еще одно решение от компании Atlassian. Jira, в отличие от Trello, является специализированным инструментом для применения гибких технологий разработки [9]. Обладает системой настроек, что, например, позволяет добавлять интеграции с репозиториями. Рассчитана на использование профессиональными командами и распространяется по подписочной модели. Основные экраны представлены на рисунках 3, 4.

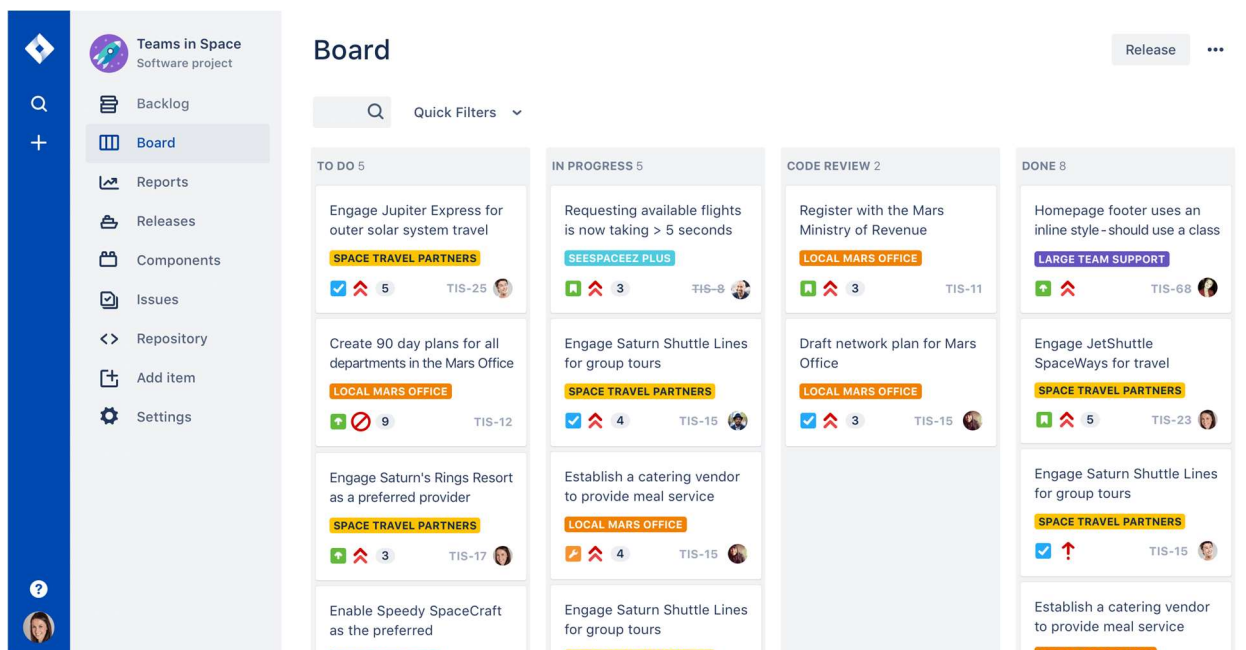


Рисунок 3 – Канбан-доска в Jira

Jira Software									
Dashboards Projects Issues Boards Structure Portfolio + Create									
Add Automation Share Export									
ALM Works PMO									
Key	Lead	Summary	Progress	Σ Budget '16	Σ Cost	Σ Time Spent	Current Status	Health	
INI-1		Core Products	<div></div>	618	275	21w 3d 7h	Addressing scope change	At Risk	
INI-3		Structure	<div></div>	499	208	16w 2d 4h	Roadmap published	At Risk	
		Roadmap Features	<div></div>	414	175	16w 2d 4h		Normal	
STR-3		Formulas	<div></div>	147	69		All ok - plan to release soon	Normal	
STR-6		As a formula author, I want to edit large formulas	<div></div>	53	14			Great	
STR-7		UX Design: larger dialog for editing formula	<div></div>	13				Great	
STR-8		Implement new UX	<div></div>	13				Great	
STR-15		As a formula author, I want to be able to use JQL	<div></div>	81	55			Normal	
STR-2		Synchronize Attribute to Custom Field	<div></div>	131	62	6w	Need to review architecture	At Risk	
STR-4		Non-Jira Notes Field / Column	<div></div>	123	44	10w 2d 4h		Normal	
		Other Features	<div></div>	72	33			Great	
INI-4		Structure Platform	<div></div>	106	67	5w 1d 3h	API changes needed	Normal	
INI-2		Extensions	<div></div>	486	303	7w 4d 7h		Great	
INI-5		Structure.Gantt	<div></div>	171	105	7w 4d 7h	Collect more use cases for	At Risk	
INI-6		Structure.Testy	<div></div>	108	43		Dark feature ready for relea	Great	
INI-7		Structure.Pages	<div></div>	194	155		Discuss the multi-instance	Great	
Showing 34 items									

Рисунок 4 – Список задач в Jira

nTask и Asana – решения, в целом аналогичные Jira как функционально, так и в плане пользовательского интерфейса [10, 11]. Однако имеют ограничение на количество пользователей в бесплатной версии. Их интерфейс представлен на рисунках 5, 6.

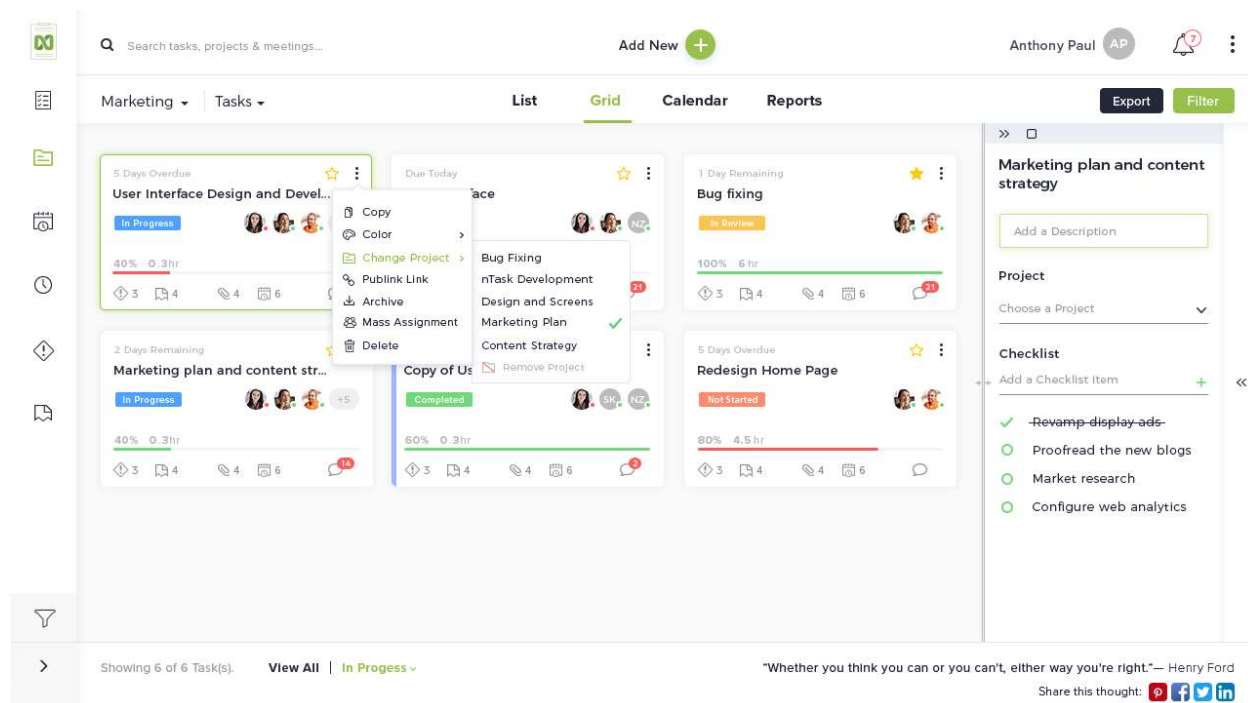


Рисунок 5 – Интерфейс nTask

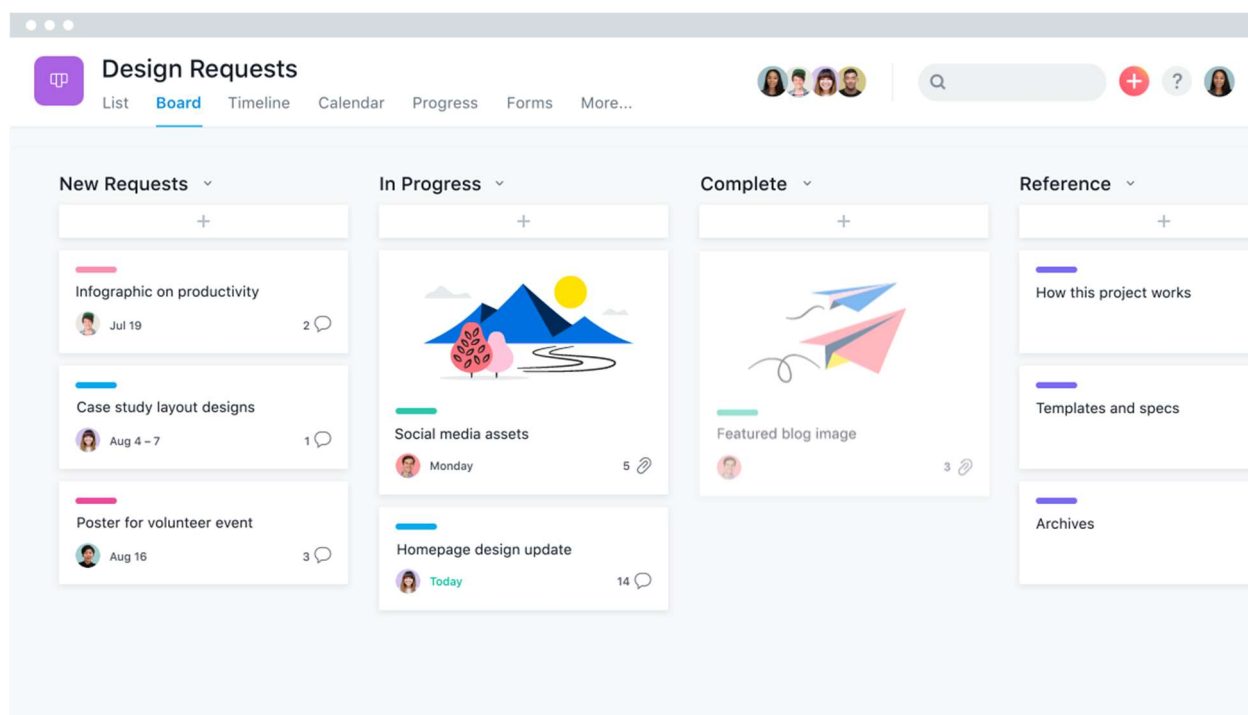


Рисунок 6 – Интерфейс Asana

Можно заметить, что все приведенные выше аналоги используют канбан-доски. Однако это не предполагается методологией и используется лишь в качестве дополнения к существующим принципам. Принципы канбан и Scrum хоть и не противоречат друг другу, но их использование вместе может привести к неправильному применению методологии.

1.4 Сравнительный анализ аналогов

Исследовав варианты использования системы управления, были составлены требования. Проведем сравнительный анализ.

Для проведения сравнительного анализа программных продуктов были выбраны следующие критерии:

а) функциональные возможности:

- добавление задач и их редактирование;
- планирование спринтов;
- наличие у задач свойств: название, описание, оценка времени, затраченное время, комментарии;
- наличие системы личных сообщений;

б) необязательные функциональные возможности:

- наличие редактируемого профиля пользователя;
- возможность установить изображение профиля;
- управление файлами;

в) технические возможности:

- хранение данных в базе данных;
- возможность внесения изменений;

г) эргономика:

- адаптивность к различным размерам экрана устройств [12];
- кроссбраузерность [13];
- интуитивно-понятный интерфейс;
- интерфейс на русском языке.

Проанализировав бизнес-процессы, возникающие во время использования систем управления, были определены коэффициенты значимости критериев. Значения этих коэффициентов представлены в таблице 1.

Таблица 1 – Критерии оценки систем бронирования

№	Наименование	Коэффициент значимости
	Функциональные возможности	
1	Добавление задач и их редактирование	1
2	Планирование спринтов	0,75
3	Наличие у задач свойств: название, описание, оценка времени, затраченное время, комментарии	0,75
4	Наличие системы личных сообщений	0,5
	Необязательные функциональные возможности	
5	Наличие редактируемого профиля пользователя	0,5
6	Возможность установить изображение профиля	0,5
7	Управление файлами	0,5
	Технические возможности	
8	Хранение данных в базе данных	0,35
9	Возможность внесения изменений	0,75
	Эргономика	
10	Адаптивность к различным размерам экрана устройств	0,5
11	Кроссбраузерность	0,5
12	Интуитивно-понятный интерфейс	0,5
13	Интерфейс на русском языке	0,5

Для оценки соответствия описанных выше критериев используется пятибалльная шкала от 0 до 4, где 4 – полное соответствие, 0 – полное несоответствие. Подробное описание шкалы представлено в таблице 2.

Таблица 2 – Описание оценок на шкале

Балл	Характеристика оценки
0	функция или качество отсутствуют и не могут быть добавлены
1	функция или качество отсутствуют, но имеется возможность их добавления
2	присутствует аналог требуемой функции
3	функция или качество реализованы, но с небольшими отклонениями от требуемых
4	функция или качество реализованы полностью

Для анализа были отобраны следующие системы:

- 1) Trello;
- 2) Jira;
- 3) LeanKit;
- 4) Asana;
- 5) nTask;
- 6) RedMine;

Оценка приведенных выше систем представлена в таблице 3. В таблице 4 приведены расчеты аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{\text{ип}} = \sum_{i=1}^n (Z_i \cdot K_i), \quad (1)$$

где:

$O_{\text{ип}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – оценка соответствия (от 0 до 4);

K_i – коэффициент значимости критерия сравнения (от 0 до 1).

Таблица 3 – Критериальная оценка программных продуктов

№	Критерий	Программный продукт №					
		1	2	3	4	5	6
1	Добавление задач и их редактирование	4	4	4	4	4	4
2	Планирование спринтов	1	4	1	4	4	3
3	Наличие у задач свойств: название, описание, оценка времени, затраченное время, комментарии	3	4	2	4	4	4
4	Наличие системы личных сообщений	1	1	0	0	0	0
5	Наличие редактируемого профиля пользователя	3	1	1	1	1	1
6	Возможность установить изображение профиля	4	4	4	4	4	4
7	Управление файлами	3	4	2	4	4	4
8	Хранение данных в базе данных	4	4	4	4	4	4
9	Возможность внесения изменений	4	4	3	4	4	4
10	Адаптивность к различным размерам экрана устройств	4	3	2	4	4	3
11	Кроссбраузерность	4	4	3	4	3	3
12	Интуитивно-понятный интерфейс	3	0	2	0	0	0
13	Интерфейс на русском языке	4	3	1	3	3	2

Таблица 4 – Итоговые оценки

№	Критерий	Программный продукт №					
		1	2	3	4	5	6
1	Оценка функциональных возможностей	7,5	10,5	6,25	10	10	9,25

Продолжение таблицы 4

№	Критерий	Программный продукт №					
		1	2	3	4	5	6
2	Оценка необ. Функциональных возможностей	5	4,5	3,5	4,5	4,5	4,5
3	Оценка технических возможностей	4,4	4,4	3,65	4,4	4,4	4,4
4	Оценка эргономики	7,5	5	4	5,5	5	4
	Интегральная оценка	24,4	24,4	17,4	24,4	23,9	22,15

Рассмотрев полученные варианты, можно составить диаграмму для сравнения интегральных оценок, представленную на рисунке 7.

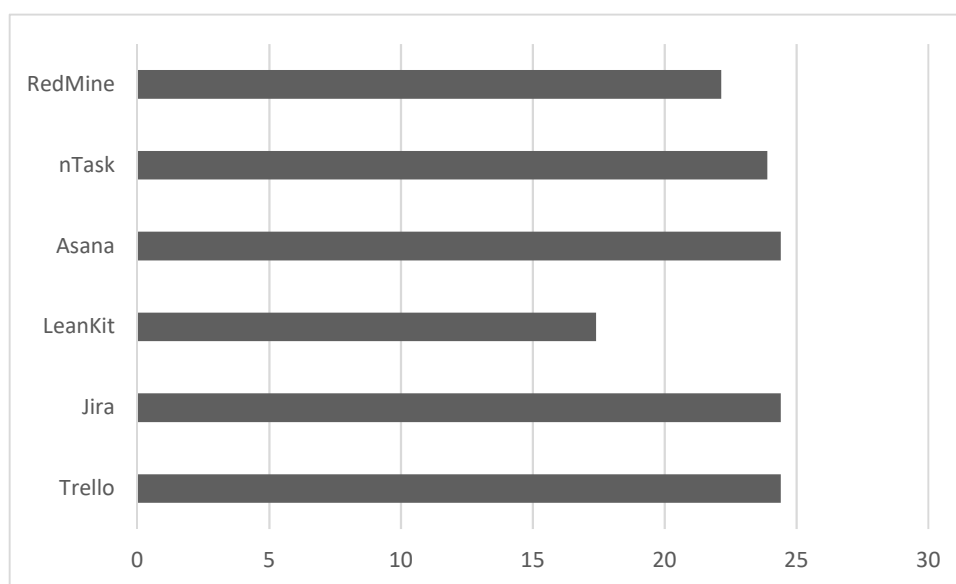


Рисунок 7 – Диаграмма интегральной оценки

Подводя итог сравнения можно сделать вывод, что лидерами являются Jira и Trello, даже несмотря на свою направленность на другой вариант использования. Аналогичные системы отстают лишь немного.

1.5 Анализ вариантов использования

Для того чтобы определить протекающие процессы в системе управления необходимо провести анализ наиболее вероятных вариантов

использования. Предметом данного исследования является система управления разработкой ПО по методологии Scrum [14].

Наиболее удобным форматом такой системы является веб-приложение, так как не требует установки и позволяет получить доступ к системе с любого устройства с установленным веб-браузером. Пользователь может зайти на сайт и произвести необходимые действия – просмотреть статус выполнения задач, добавить новые задачи и т.д.

Одной из особенностей Scrum является то, что все участники команды разработчиков имеют одинаковые права, поэтому пользователю достаточно ввести данные для входа в свою ученую запись и таким образом получить доступ ко всем возможностям приложения. Среди этих возможностей: создание и редактирование задач, создание и редактирование спринтов, просмотр состояния спринтов.

Рассмотрим особенности задач в системе управления. Единственная особенность, которая подразумевается в используемой методологии – задача должна быть понятна и выполнима за небольшой промежуток времени (не более 4 недель). Разработчикам придется самим следить за выполнением этого правила. Для удобства задача должна иметь следующие характеристики:

- номер;
- название;
- описание;
- создатель задачи;
- назначенный исполнитель;
- оценка задачи по времени выполнения;
- затраченное время на выполнение;
- спринт, в котором находится задача.

Второй важной функцией системы является создание спринтов. Спринт является множеством задач, которые должны быть выполнены за установленный промежуток времени (1-4 недели), и в результате выполнения

которых должен получиться продукт, готовый к потенциальному выпуску на рынок. Спринт определяется следующими характеристиками:

- номер;
- краткое описание целей;
- задачи, включенные в спринт.

В конце отведенного периода времени команда оценивает свою работу и помечает спринт как завершённый. Таким образом, спринт может находиться в трех состояниях:

- будущий;
- текущий (только один одновременно);
- прошедший.

Проанализировав приведенные выше особенности, можно перейти к определению вариантов использования автоматизированной информационной системы, обслуживающей предполагаемых пользователей.

Информационные процессы, которые возникают во время работы с сайтом системы бронирования достаточно простые и сводятся к небольшому набору основных действий:

- просмотр бэклога;
- просмотр и редактирование задач;
- просмотр и редактирование спринтов;
- настройка нового пользователя;
- аутентификация и авторизация пользователя.

Рассмотрим варианты использования подробнее. Зайдя на сайт, система управления предлагает ввести электронную почту и пароль. Без аутентификации пользователь не имеет доступа к другим частям приложения. После успешного входа в систему пользователю предоставляется весь доступный функционал. Рассмотрим подробно все возможные варианты использования.

Настройка нового пользователя. Так как приложение предполагает размещение серверной части на компьютере заказчика, то для максимальной

безопасности добавление нового пользователя происходит путем добавления основных данных (почта и имя) с помощью запроса к базе данных администратором системы. Затем при первой попытке входа пользователю будет предложено создать пароль для своей учетной записи. После создания пароля пользователь сможет самостоятельно редактировать свои данные.

Аутентификация и авторизация. Пользователь вводит данные для доступа (электронная почта и пароль). Система проверяет полученные данные, сравнивая их с хранящимися в БД. При нахождении соответствия пользователь успешно заходит в свою учетную запись и получает доступ к системе, в противном случае – получает соответствующее сообщение об ошибке. Так же после входа пользователь может видеть свой профиль и редактировать его при необходимости, а также видеть профили других участников команды.

Просмотр и редактирование задач. Доступ к задачам и к CRUD операциям над ними (create – создание, read – просмотр, update – редактирование, delete – удаление) предоставляется только авторизованным пользователям. Если пользователь не авторизован, но пытается каким-либо образом получить доступ к задаче или пытается получить доступ к несуществующей задаче, будучи авторизованным, то сервер должен отправлять ответ с соответствующим кодом ошибки (статусы 403 и 404).

Просмотр и редактирование спринтов. Доступ к спринтам также предоставляется только авторизованным пользователям. И также, как и в случае с задачами, сервер должен отправлять сообщение с кодом ошибки при необходимости.

Описав и проанализировав все возможные варианты использования приложения, можно составить UML-диаграмму [15] вариантов использования, отображающую возможности приложения (рисунок 8).

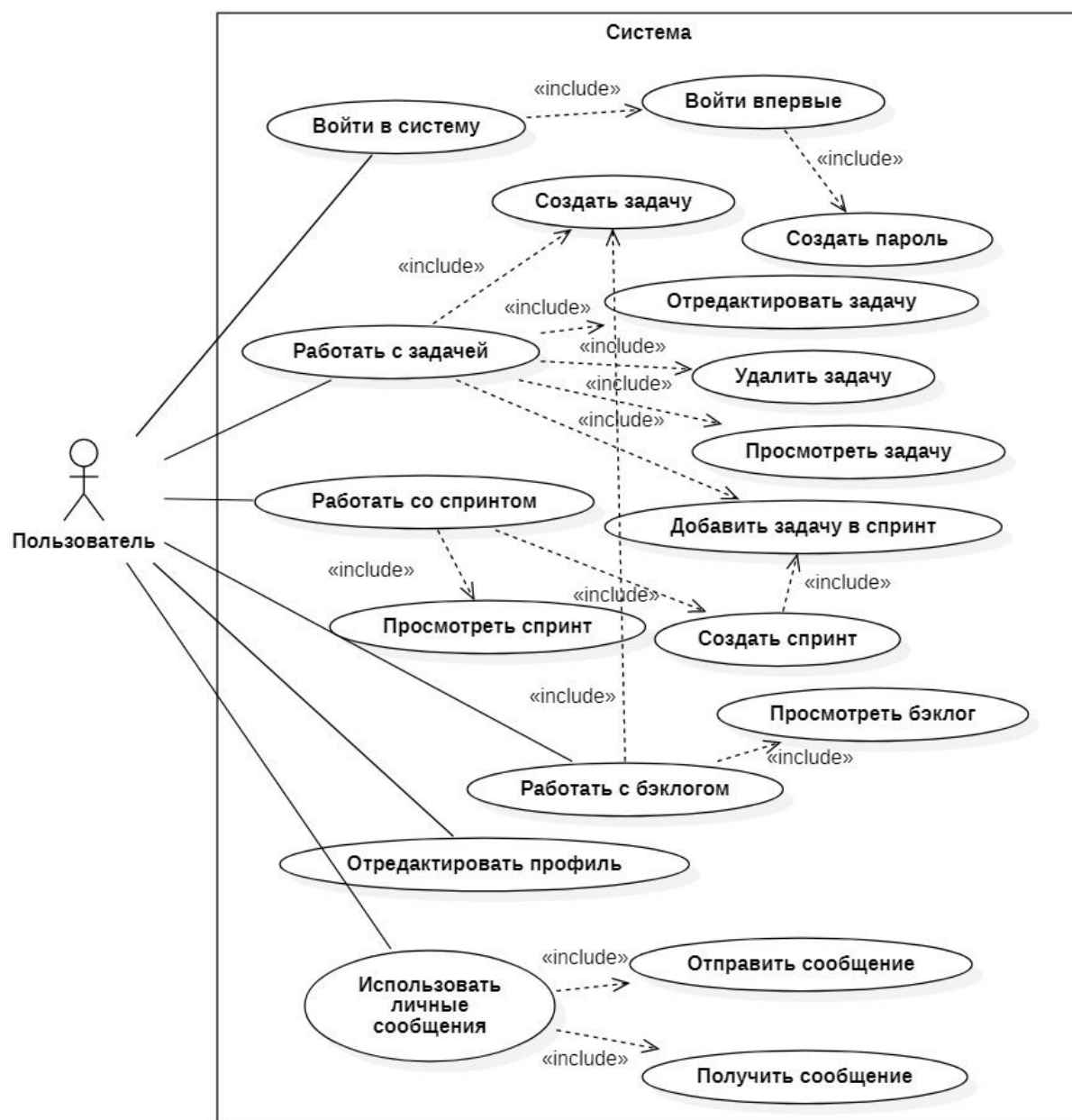


Рисунок 8 – Диаграмма вариантов использования

Проанализировав диаграмму вариантов использования, можно составить диаграмму деятельности. Для этого рассмотрим один из сценариев использования приложения. Пользователь входит в систему и выполняет некоторые действия. Диаграмма деятельности описанного сценария представлена на рисунке 9.

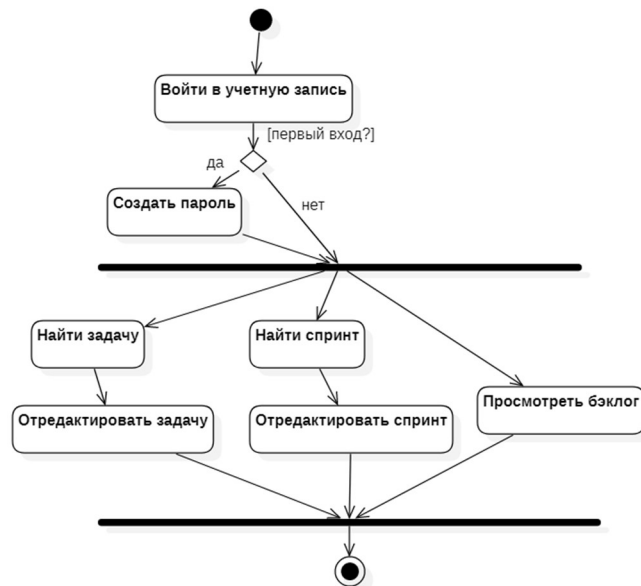


Рисунок 9 – Диаграмма деятельности

На основе вариантов использования, диаграммы прецедентов и диаграммы деятельности была сформирована последовательность действий для следующих прецедентов:

- 1) редактирование задачи (рисунок 10);



Рисунок 10 – Диаграмма последовательности действий для прецедента
«Редактирование задачи»

2) создание спринта (рисунок 11).



Рисунок 11 – Диаграмма последовательности действий для прецедента
«Создание спринта»

Проанализировав варианты использования системы управления, можно перейти к определению структурной модели сайта.

1.6 Структура приложения

Поскольку разрабатываемая система должна быть максимально простой в установке и эксплуатации, то пользовательский интерфейс будет реализован в виде веб-сайта. Такой вариант не требует установки и наличия особой конфигурации пользовательской системы.

На основе, составленной при анализе вариантов использования UML диаграммы, можно составить общую структуру сайта, схема которой представлена на рисунке 12.

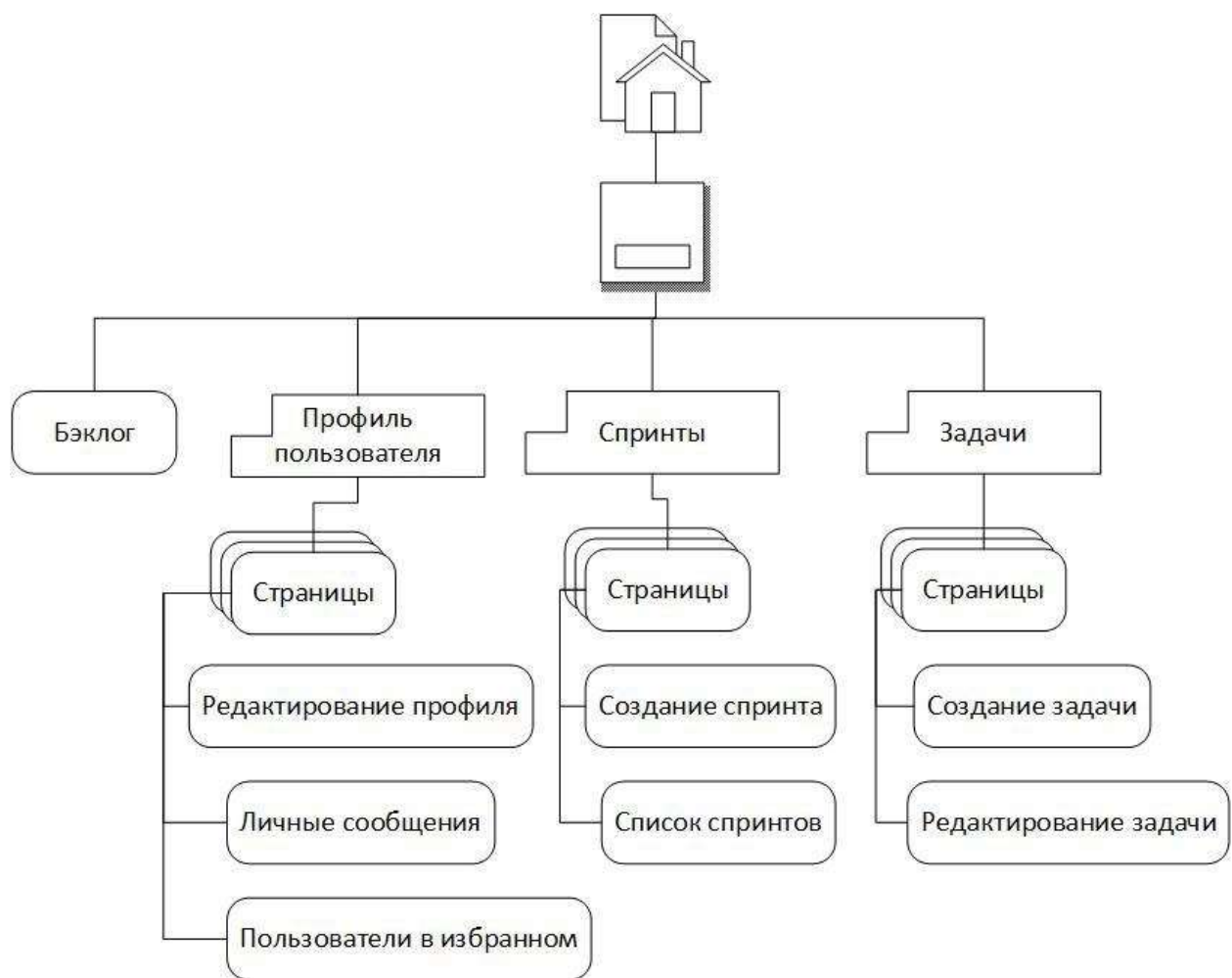


Рисунок 12 – Общая структура сайта

Смоделировав структуру сайта, можно перейти к составлению требований к системе управления контентом и подсистеме бронирования.

1.7 Техническое задание

Назначение разработки:

- создание и отслеживание статуса задач для разработки;
- предоставление инструментов для планирования спринтов.

Цель создания веб-приложения: предоставление пользователям вспомогательного инструмента для применения методологии Scrum.

1.7.1 Функциональные требования к программному продукту

Разрабатываемая система управления обязательно должна иметь следующие характеристики и функции:

- обеспечение доступа к системе только для авторизованных пользователей;
- управление задачами и их свойствами (добавление, просмотр, изменение, удаление);
- управление спринтами и их свойствами (добавление, просмотр, изменение, удаление);
- возможность редактирования своего профиля и просмотра профилей других пользователей;
- возможность писать и получать личные сообщения от других пользователей.

1.7.2 Требования к графическому дизайну сайта

Основные разделы сайта должны быть доступны с любой страницы. На главной странице сайта необходимо разместить информацию о текущих задачах пользователя. Сайт должен успешно открываться во всех популярных браузерах и на различных размерах экрана.

1.7.3 Требования к разделению доступа

Так как все пользователи системы имеют одинаковые права, то система должна предоставлять доступ ко всем функциям системы для каждого авторизованного пользователя. Если же пользователь пытается получить доступ без авторизации, то на каждый его запрос (кроме запроса авторизации) должен быть отправлен ответ со статусом 403 или 401.

1.7.4 Требования к хранению данных

Всю информацию и данные сайта необходимо хранить в структурированном виде в реляционной БД под управлением СУБД. Файлы изображений, видео, аудио и документов, которые предназначены для просмотра и скачивания, необходимо хранить в облаке для облегчения

нагрузки на базу данных. Наиболее предпочтительно использование СУБД, использующих язык SQL.

1.7.5 Требования к языкам программирования и разметки

Для реализации статических элементов страниц и шаблонов необходимо использовать язык разметки HTML 5 и CSS3. Исходный код желательно разрабатывать по стандартам W3C. Для реализации интерактивных элементов клиентской части необходимо использовать язык программирования TypeScript, транслируемый в JavaScript, и CSS3 анимацию. Для реализации серверной части предпочтительно использование языка программирования C# и фреймворков ASP.NET Core и Entity Framework Core.

1.7.6 Требования к иллюстрациям

Все изображения и фотографии, составляющие часть контента (кроме элементов дизайна страницы) необходимо сопровождать альтернативным текстом. Желательно использовать рисунки в распространенных форматах jpg и gif.

1.7.7 Требования к программному обеспечению серверной части

Для успешной работы сайта требуется следующее программное обеспечение:

- операционная система Windows 10, Windows Server 2016 и позже, macOS или Linux;
- веб-сервер IIS, nginx, Apache, Azure или любой другой с поддержкой .NET Core приложений;
- SQLite;
- платформа .NET.

1.7.8 Требования к аппаратному обеспечению

Для нормального функционирования сайта требуются следующее техническое обеспечение со следующими минимальными характеристиками:

- процессор Intel Core i3;
- оперативная память – не менее 8096 МБ;
- объем накопителя – не менее 15 ГБ свободного места.

1.7.8 Требования к клиентскому программному обеспечению

Верстка сайта должна быть кроссбраузерной, сам сайт должен быть доступен для полнофункционального просмотра при помощи следующих браузеров:

- Microsoft Edge 80 и выше;
- Opera 65 и выше;
- Firefox 70 и выше;
- Google Chrome 80 и выше.

1.7.9 Требования к лингвистическому обеспечению

Текстовое содержание и элементы интерфейса сайта должны быть выполнены на русском языке.

1.7.10 Требования к эргономике и технической эстетике

Сайт должен быть оптимизирован для просмотра при разрешении окна просмотра 1280 на 960 и выше без горизонтальной полосы прокрутки. Элементы управления должны быть сгруппированы однотипно на всех страницах.

Графический интерфейс подключаемых модулей необходимо выполнять в едином стиле с общим дизайном. Он должен обеспечивать возможность логичного перемещения между составляющими системы. Для выполнения однотипных операций необходимо использовать одинаковые процедуры управления и навигационные элементы, тем самым создавая интуитивно понятный интерфейс.

1.7.11 Требования к персоналу

Для работы с веб-интерфейсом системы управления содержимым от администратора не должно требоваться специальных навыков и знаний, кроме общих навыков работы с персональным компьютером и стандартным браузером.

2 Планирование разработки и оценка бюджета

2.1 Календарное и ресурсное планирование

Неотъемлемой частью проектирования и разработки автоматизированной системы является календарное и ресурсное планирование проекта. Планирование позволяет осуществлять управление ходом работ по созданию информационной системы.

Календарный план проекта содержит определение длительности работ и их взаимосвязей. Диаграмма Ганта позволяет в удобной графической форме представить сроки работ по проекту, продолжительность этапов, список требуемых ресурсов и т.д.

Кроме перечня работ в календарное планирование входит составление ресурсной модели проекта. Среди всех ресурсов можно выделить несколько основных категорий: трудовые, материальные и невозобновляемые. Ресурсы по-своему влияют на их использование и затраты, которые и составляют бюджет проекта.

Для получения плана-графика проекта с оптимальным использованием ресурсов и минимальными сроками проведем календарное планирование.

Составим план стадий и этапов создания веб-приложения. План, содержащий подробную информацию о наименовании и длительности стадий и этапов разработки, а также лиц, выполняющих их, представлен в Приложении Н.

Для составления плана будет использоваться программа Microsoft Project. Создадим диаграмму Ганта и внесем в нее планируемые работы и этапы для создания продукта (приложение П).

А также внесем информацию о трудовых и материальных ресурсах (рисунок 13).

Название ресурса ▼	Тип ▼	Единицы измерения материалов ▼	Краткое название ▼	Группа ▼	Макс. единиц ▼	Стандартная ставка ▼
Руководитель проекта	Трудовой		РП		100%	520.00 Р/час
Программист (backend)	Трудовой		ПБ		100%	500.00 Р/час
Программист (frontend)	Трудовой		ПФ		100%	450.00 Р/час
Тестировщик	Трудовой		Т		100%	390.00 Р/час
Дизайнер	Трудовой		Д		100%	350.00 Р/час
Компьютер	Материальный		К			120,000.00 Р
Принтер	Материальный		П			20,000.00 Р
Бумага	Материальный		Б			450.00 Р
Интернет	Материальный		И			2,000.00 Р

Рисунок 13 – Лист ресурсов проекта

Проанализировав этапы и их продолжительность можно сделать вывод, что на проектирование и разработку рассматриваемого программного продукта потребуется 376 часов.

2.2 Расчет бюджета

Проведем расчет стоимости проектирования и разработки проекта.

Для расчета затрат необходимо учесть следующие статьи расходов:

- амортизация;
- материалы;
- фонды основной и дополнительной заработной платы;
- социальные выплаты;
- прочие затраты.

Амортизация

Амортизация – это процесс перенесения по частям стоимости изношенной части основных производственных фондов (ОПФ) на стоимость готовой продукции.

Стоимость ОПФ и расчет амортизационных отчислений приведены в таблице 14. Обозначения в таблице:

- СПИ: срок полезного использования;
- НА: норма амортизации;
- А: амортизация;

– ТЗР: транспортно-заготовительные расходы.

Таблица 14 – Амортизационные отчисления

Оборудование	Стоимость, руб.	СПИ, лет	НА, %	А, год, руб.
Компьютер с ПО: MS Office 2019; MS Windows 10	120 000	5	20	24 000
Принтер Brother	20 000	5	20	4 000
ТЗР (1%)	550			
Итого	141 400			28 000

Для разработки программного продукта планируется использовать программное обеспечение в свободном доступе: Visual Studio Code, Visual Studio 2019, SQLite, веб-сервер IIS, JavaScript фреймворк – Angular [16].

Амортизационные отчисления рассчитываются по следующей формуле:

$$A = C_{\text{пер}} * НА, \quad (1)$$

где:

А – амортизационные начисления, руб.;

$C_{\text{пер}}$ – первоначальная стоимость основных средств, руб.;

НА – норма амортизации, которая находится по формуле:

$$НА = 1 / СПИ, \quad (2)$$

где:

СПИ – срок полезного использования основных средств.

Срок полезного использования компьютера = 5 лет

$$НА_{\text{комп}} = 1 / 5 = 0,2$$

Срок полезного использования принтера = 5 лет

$$НА_{\text{принт}} = 1 / 5 = 0,2$$

$$A_{\text{комп}} = 120\,000 * 0,2 = 24\,000 \text{ руб.}$$

$$A_{\text{принт}} = 20\,000 * 0,2 = 4000 \text{ руб.}$$

$$A_r = 24\,000 + 4000 + 550 = 28\,550 \text{ руб.}$$

$$A_m = 28\,550 / 12 = 2379,2 \text{ руб/мес.}$$

$$A_{\text{итог}} = 2379,2 * 1 = 2379,2 \text{ руб.}$$

Работа по созданию программного продукта предполагается вести 376 часов, расчёт ведётся за 1 месяц.

Материалы

В данной статье приведены расчеты оборотных средств, использовавшиеся для разработки программного продукта, они приведены в таблице 15.

Таблица 15 – Затраты по статье «Материалы»

Наименование	Количество, шт.	Сумма, руб.
Бумага	2	900
ТЗР (1%)		10
ИТОГО:		910

Фонд основной и дополнительной заработной платы

Зарплата – вознаграждение за труд в зависимости от квалификации работника, сложности, количества, качества и условий выполняемой работы.

Трудоемкость всех видов работ, связанных с разработкой рассматриваемого программного продукта представлена в таблице 15.

Таблица 15 – Расчет основной и дополнительной заработной платы

Исполнитель	Трудоемкость, часов (Тр)	Ставка, руб./час	ОЗП, руб.	ДЗП, руб.	Соц. выплаты
Руководитель проекта	80	520	41 242,40	4 124,24	13 609,99
Программист (frontend)	112	500	55 191,50	5 519,15	18 213,20
Программист (backend)	112	450	49 247,80	4 924,78	16 251,78
Тестировщик	32	390	12 372,60	1 237,26	4 082,78
Дизайнер	40	350	13 646,25	1 364,63	4503,27
Итого:	376		171 700,55	17 170,06	56 661,02

На разработку и тестирование программного продукта затрачено 376 часов.

Рассмотрим оплату труда на примере расчета заработной платы руководителя проекта.

Оплата за труд рассчитывается следующим образом:

$$\text{ОЗП} = \text{Сч} * \text{Тр}, \quad (3)$$

где:

Тр – трудоемкость.

Сч – тарифная ставка, которая рассчитывается по формуле:

$$\text{Сч} = (\text{МРОТ} * \text{Тк}) / 160, \quad (4)$$

где:

МРОТ – минимальный размер оплаты труда = 12 130 руб.;

Тк – тарифный коэффициент;

160 – количество часов в месяц по ТК РФ.

$$\text{Тк}_{\text{рук}} = 6,8$$

$$\text{Сч}_{\text{рук}} = (12\,130 * 6,8) / 160 = 515,53$$

$$\text{ОЗП}_{\text{рук}} = 515,53 * 80 = 41\,242,40 \text{ руб.}$$

Мотивация труда оценивается через дополнительную заработную плату, которая рассчитывается по формуле:

$$\text{ДЗП} = 10\% \text{ ОЗП}, \quad (5)$$

$$\text{ДЗП}_{\text{рук}} = 10\% * 41\,242,40 = 4\,124,24 \text{ руб.}$$

Социальные выплаты

Социальные выплаты состоят из:

– «Фонд социального страхования» (ФСС) – 5,9%;

– «Пенсионный фонд РФ» (ПФР) – 22%;

– «Фонд обязательного медицинского страхования» (ФОМС) – 2,1%.

$$\text{СВ} = 30\% (\text{ОЗП} + \text{ДЗП}) \quad (6)$$

$$\text{СВ}_{\text{рук}} = 30\% * (41\,242,40 + 4\,124,24) = 13\,609,99 \text{ руб.}$$

Прочие

В этой статье приведены затраты на доступ в Интернет и электроэнергию (ЭЭ)

Затраты на электроэнергию рассчитываются по формуле:

$$\text{ЭЭ} = T_c * W * T_{\text{раб}}, \quad (7)$$

где:

T_c – стоимость 1 кв/ч = 3,63 руб.

W – энергомощность компьютера = 350 Вт/ч = 0,35 кВт/ч

$T_{\text{раб}}$ – время создания, т.к. программный продукт разрабатывался 376 часов

$$\text{ЭЭ} = 3,63 * 0,35 * 568 = 721,64 \text{ руб.}$$

Стоимость услуги доступа в Интернет:

Работа над программным продуктом велась 376 часов, тариф за услугу доступ в Интернет составляет = 2000 руб./мес. Однако, провайдер не предоставляет почасовой доступ в интернет, поэтому расчет ведется за 2 месяца.

$$\text{Интернет} = 2000 * 2 = 4000 \text{ руб.}$$

$$\text{Прочие затраты} = \text{ЭЭ} + \text{Интернет} = 721,64 + 4000 = 4721,64 \text{ руб.}$$

Структура затрат представлена в таблице 16.

Таблица 16 – Структура затрат

Наименование статей	Затраты, руб.	%
Амортизация	2379,2	0,9%
Материалы	910	0,4%
Фонд основной и дополнительной ЗП	188 870,61	74,5%
Социальные выплаты	56 661,02	22,3%
Прочие	4721,64	1,9%
Затраты на проект	253 542,47	

Диаграмма структуры затрат представлена на рисунке 14.

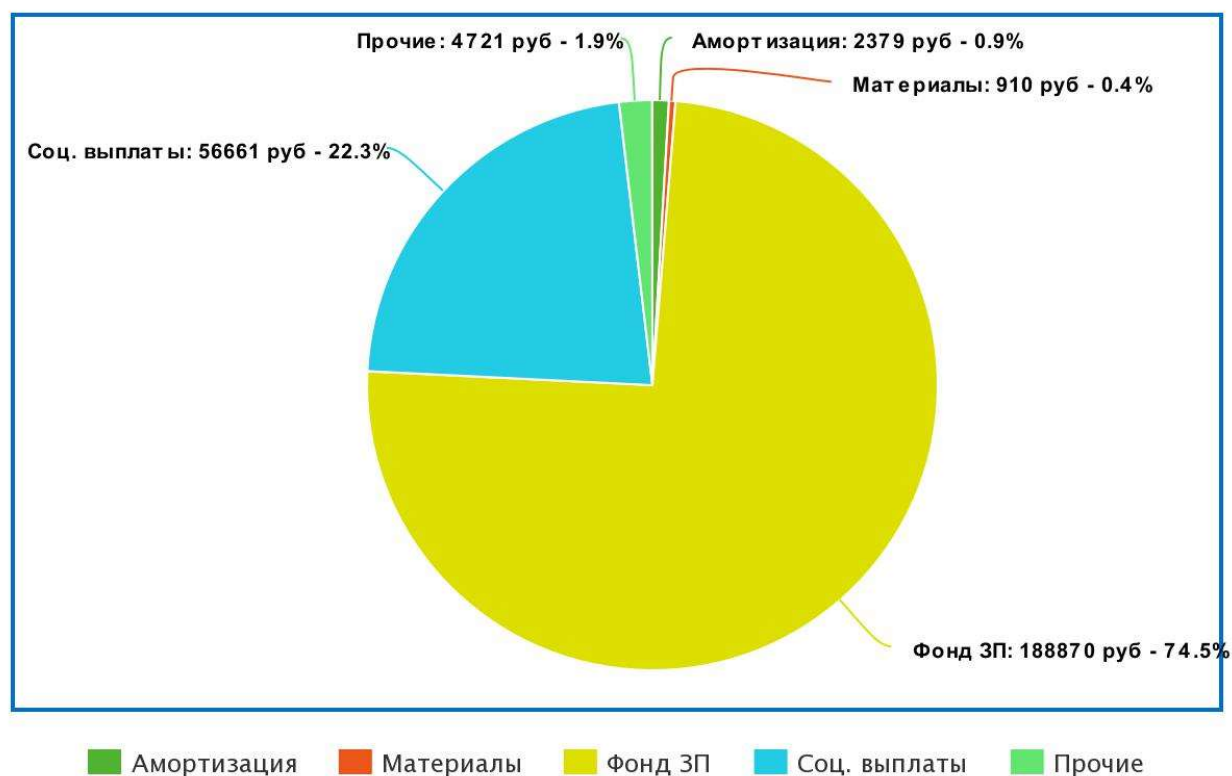


Рисунок 14 – Структура затрат

Проводя анализ структуры расходов можно сделать вывод, что основными статьями расходов является фонд оплаты труда и социальные выплаты. Затраты на проектирование и разработку программного продукта «Веб-приложение для организации управления разработкой программных проектов по методологии SCRUM» составят 253 542,47 рублей.

2.3 Выбор программно-аппаратных средств

Веб-приложение представляет собой клиент-серверное приложение, в котором все взаимодействие происходит в браузере [17]. Главные преимущества приложения в браузере – отсутствие необходимости установки и обновления клиента, а также доступность на любом устройстве с веб-браузером независимо от программно-аппаратных средств. Однако могут возникнуть ошибки из-за различных реализаций JavaScript и CSS в браузерах, поэтому необходимо проверять работоспособность в нескольких браузерах.

Сайт – это одна или несколько логически связанных между собой страниц, расположенных на веб-сервере и просматриваемых через веб-браузер [18, 19].

Динамический сайт – это сайт, состоящий из страниц, генерируемых программным обеспечением динамически. В свою очередь динамическая страница состоит из контента, шаблонов и скриптов. Таким образом, пользователь видит страницу, полученную из нескольких файлов – шаблонов, таблиц стилей, изображений и т.д. Использование шаблонов позволяет быстро и удобно генерировать однотипные страницы и элементы страницы.

Генерация содержимого на стороне клиента производится браузером после получения данных с сервера. Клиент запрашивает скрипты и шаблоны веб-страницы, которые в свою очередь делают запрос на сервер для получения данных, которыми наполнится веб-страница.

Шаблон проектирования MVC – один из наиболее подходящих для реализации описанного выше процесса. Основная идея шаблона заключается в разделении приложения на три главных компонента – модель, или данные (Model), представление, или пользовательский интерфейс (View), а также контроллер, отвечающий за логику приложения (Controller). Таким образом, компоненты получают относительно независимыми, а значит модификация каждого из них практически не сказывается на работе других [20].

Основными технологиями для создания серверной части веб-приложений на сегодняшний день являются: ASP.NET, ASP.NET Core, PHP, Django, Perl, Ruby on Rails, Node.js и Go. Рассмотрим их подробнее.

ASP.NET и ASP.NET Core – мощные фреймворки для веб-разработки от компании Microsoft. Активно поддерживаются и развиваются оба, однако Microsoft подталкивает разработчиков к переходу на ASP.NET Core, который является улучшенной кроссплатформенной версией ASP.NET [21].

PHP (Hypertext Preprocessor – препроцессор гипертекста) – популярный язык сценариев общего назначения для веб-разработки. Поддерживается

подавляющим большинством хостингов и имеет одно из самых крупных сообществ разработчиков [22].

Django – высокоуровневый фреймворк для веб-разработки на языке программирования Python. Ориентирован на скорость, безопасность и масштабируемость, а также на быструю разработку с написанием минимального количества кода [23].

Perl – интерпретируемый высокоуровневый язык программирования общего назначения [24]. Среди главных особенностей – большое количество инструментов для работы с текстом, что позволяет создавать динамические веб-страницы. Однако с 2006 года стремительно теряет свою популярность [25].

Ruby on Rails – веб-фреймворк на языке Ruby. Один из первых подобных фреймворков, оказал заметное влияние на будущие аналоги. Подразумевает использование парадигм программирования вроде DRY (Don't Repeat Yourself – «не повторяйся») и CoC (Convention over Configuration – «соглашения по конфигурации») [26, 27].

3 Проектирование системы управления

3.1 Проектирование пользовательского интерфейса

Поскольку система управления будет размещена на сайте, то и вся клиентская часть приложения является веб-сайтом. Сайт – это система из страниц. Это документ, написанный на языке разметки гипертекста (HTML) и стилизованный с помощью каскадных таблиц стилей (CSS), который можно просмотреть с помощью веб-браузеров, таких как Google Chrome, Mozilla Firefox, Opera, Safari и др.

Страницы клиентской части должны быть спроектированы таким образом, чтобы они корректно отображались во всех современных браузерах, в том числе и на мобильных устройствах. Страницы должны адаптироваться к различным размерам экрана и отображать весь контент вне зависимости от разрешения и размера экрана устройства. Интерфейс должен быть логично структурирован и интуитивно понятен пользователю. Для этого все страницы сайта должны иметь схожую структуру.

Изучив тренды современного дизайна, был спроектирован общий макет сайта. Графический интерфейс всех страниц сайта должен содержать следующие элементы:

- верхнюю часть, состоящую из логотипа с левой стороны и кнопки доступа к профилю с правой стороны;
- навигационное меню сайта также в верхней части для обеспечения доступа ко всем разделам из любого раздела;
- основную область, отображающую контент выбранной страницы сайта.

Ниже представлены макеты следующих страниц: страница авторизации (рисунок 15) и страница редактирования задачи (рисунок 16).

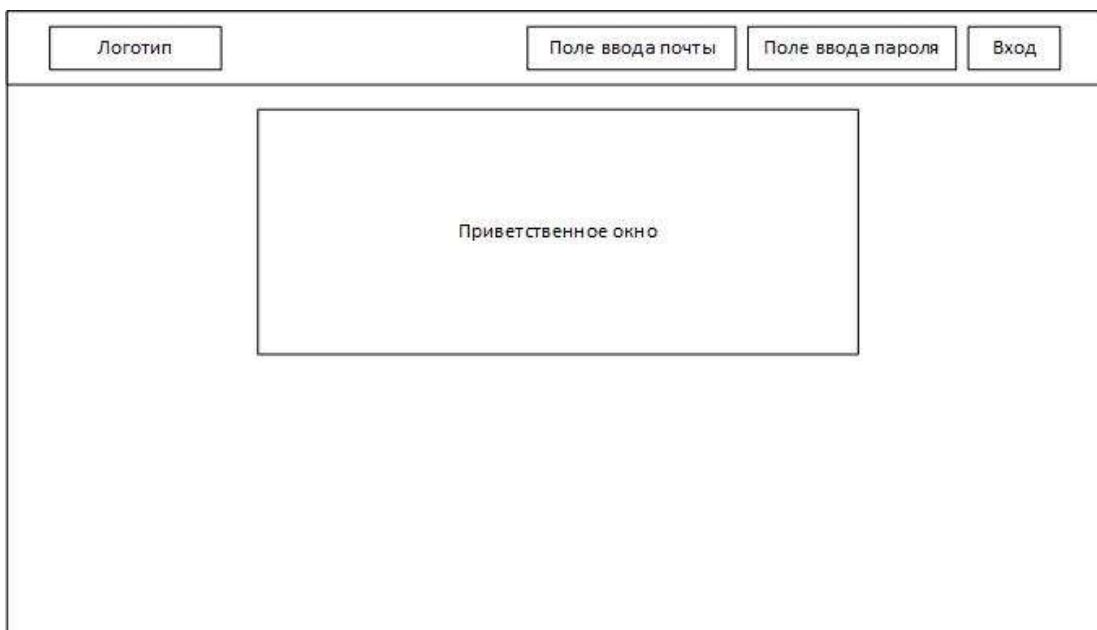


Рисунок 15 – Макет страницы авторизации

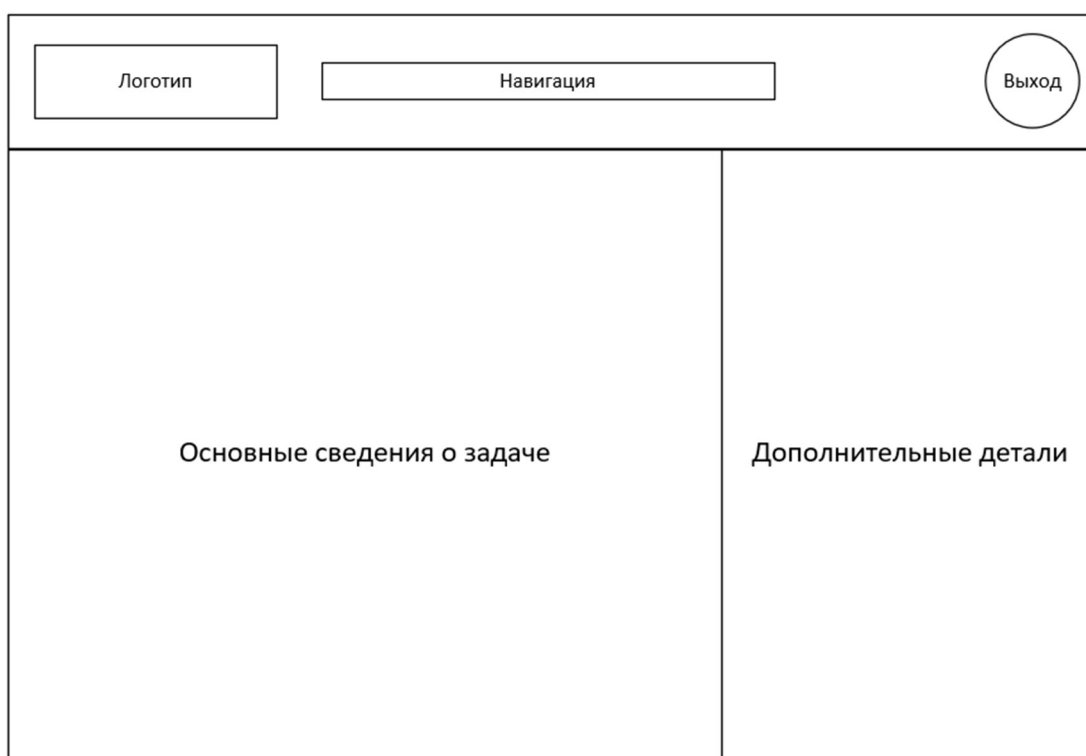


Рисунок 16 – Макет страницы редактирования задачи

Другие страницы имеют похожую структуру и могут иметь различное наполнение, однако строка навигации должна всегда оставаться наверху любой страницу, вне зависимости от каких-либо условий.

3.2 Организация процесса обработки информации

Проанализировав варианты использования и протекающие бизнес-процессы в системе управления, были определены необходимые компоненты и их взаимосвязи (рисунок 17).

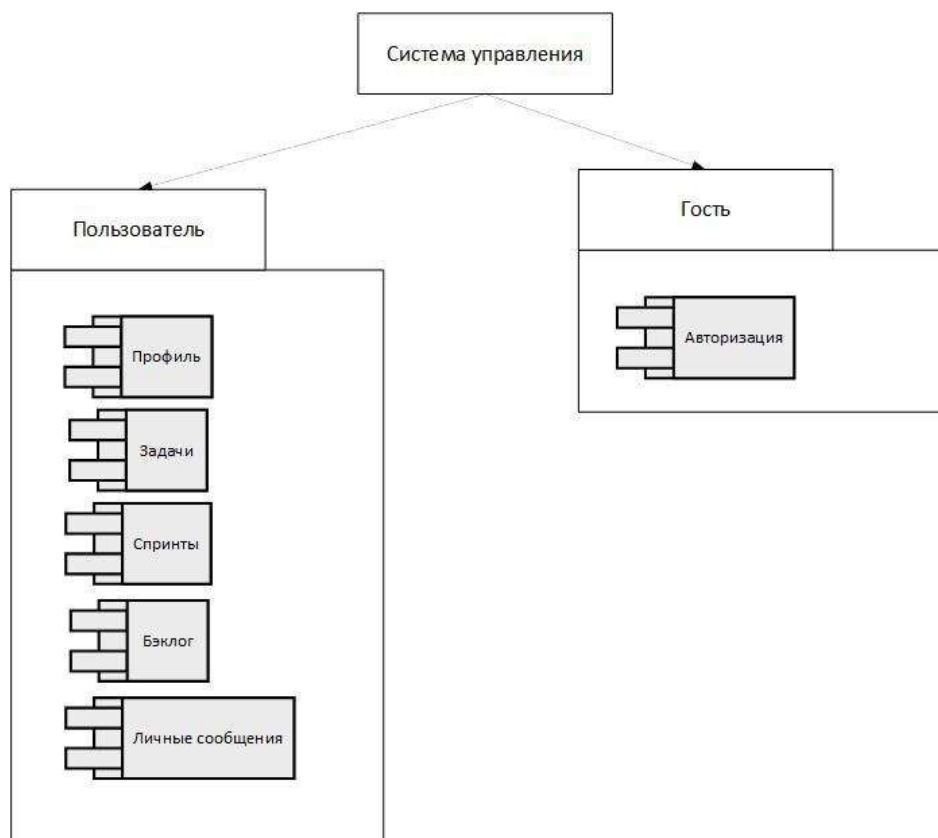


Рисунок 17 – Диаграмма компонентов

За обработку всех запросов обычно отвечает контроллер. Он получает запрошенный адрес, анализирует запрос, выделяя название метода, тип запроса, параметры и тело запроса и потом вызывает соответствующий метод, передавая ему данные из запроса.

Контроллер обеспечивает связь между пользователем и системой, получая запрос от клиента и передавая управление сервисному слою приложения. Сервисы приложения выполняют все необходимые действия по обработке данных и работе с базой данных. После завершения работы

необходимые данные возвращаются в контроллер, который в свою очередь формирует ответ для клиента в формате JSON.

Для взаимодействия с базой данных сервисы используют Entity Framework. С его помощью можно выполнять действия с базой данных на языке C#, так как он автоматически преобразует запрос к БД на языке C# в SQL запрос, который и будет применен к подключенной базе. Такой подход позволяет значительно сократить время разработки алгоритмов взаимодействия приложения с хранилищем данных.

Для авторизации же будет использоваться JWT – JSON Web Token. Смысл данного метода заключается в том, что при вводе данных для авторизации пользователь получает свой уникальный токен, который он отправляет с каждым последующим запросом, чтобы система поняла, что запрос идет именно от этого авторизованного пользователя, а не от другого или неавторизованного пользователя.

Каждый JWT токен состоит из трех частей (header – заголовок, payload – полезная нагрузка, signature – подпись), разделенных точкой. В заголовке содержится информация об алгоритме для подписи, в данном случае – «JWT». В полезной нагрузке кодируется информация о пользователе. Подпись вычисляется из заголовка, полезной нагрузки, секрета и алгоритма, указанного в заголовке.

3.3 Структуры хранения данных

Проанализировав информационные процессы, возникающие во время использования приложения, были определены основные типы и объемы обрабатываемой информации. Диаграмма базы данных представлена на рисунке 18. На диаграмме также отображены типы данных, допустимы ли значения null, являются ли столбцы первичными ключами, а также отношения внешних ключей.

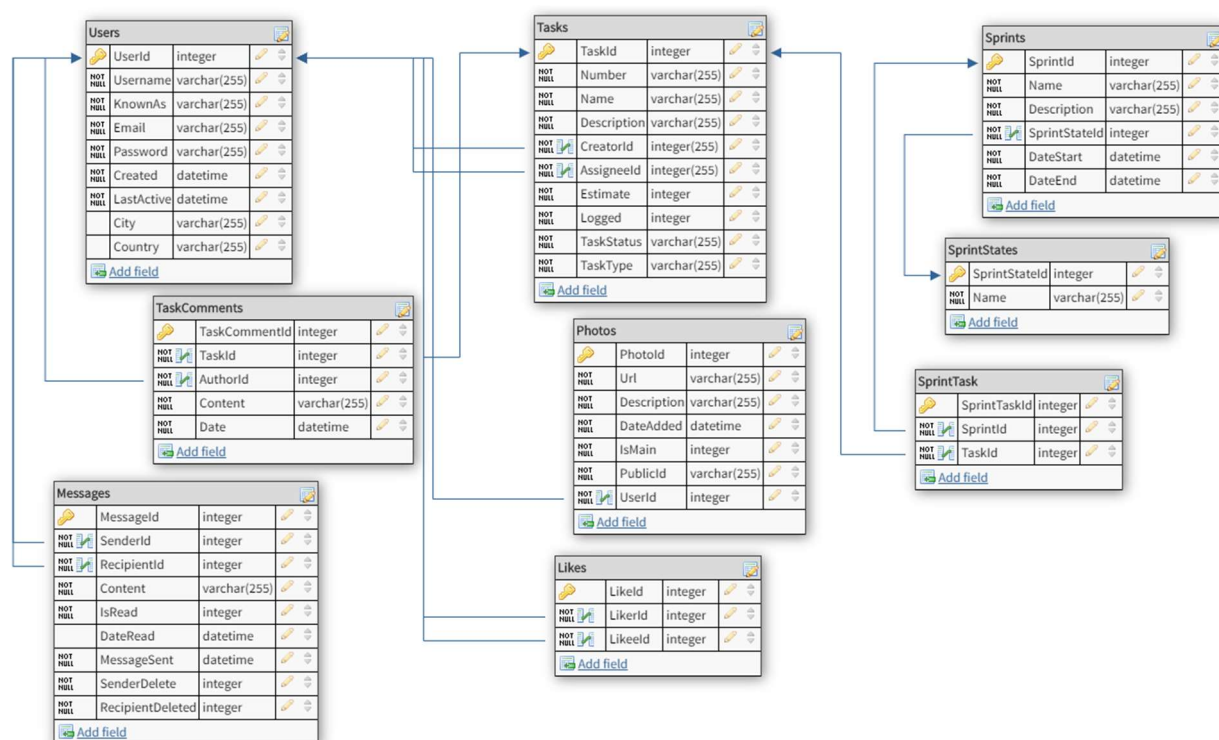


Рисунок 18 – Диаграмма классов базы данных

Для хранения всех данных системы бронирования и содержимого сайта используется SQL Server. База данных состоит из 8 таблиц. Все данные в таблице приведены в нормальную форму.

Основные данные хранятся в следующих таблицах: в таблице «Users» содержится информация о пользователях системы, а также информация для авторизации, в таблице «Tasks» – все задачи в системе, а в таблице «Sprints» – все спринты, в таблице «Photos» – данные для получения фото из облака. Вспомогательная таблица «SprintTasks» связывает спринты и задачи, так как эти сущности имеют отношение «многие-ко-многим». Таблица «TaskComments» служит для хранения комментариев к задаче.

В таблице 17 приведены характеристики таблицы «Users».

Таблица 17 – Таблица «Users» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	UserId	int
2	Метка удаления	Username	varchar(255)
3	Имя	KnownAs	varchar(255)

Продолжение таблицы 17

№	Название	Идентификатор	Тип данных, длина
4	Фамилия	Email	varchar(255)
5	Электронная почта	Password	varchar(255)
6	Пароль	Created	datetime
7	Телефон	LastActive	datetime
8	Информация	City	varchar(255)
9	Метка активности	Country	varchar(255)

В таблице 18 приведены характеристики таблицы «Tasks».

Таблица 18 – Таблица «Tasks» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	TaskId	int
2	Код	Number	varchar(255)
3	Название	Name	varchar(255)
4	Описание	Description	varchar(2048)
5	Идентификатор создания	CreatorId	int
6	Идентификатор назначенного исполнителя	AssigneeId	int
7	Временная оценка в часах	Estimate	int
8	Количество затраченных часов	Logged	int
9	Идентификатор статуса задачи	TaskStatus	varchar(255)
10	Идентификатор типа задачи	TaskType	varchar(255)

В таблице 19 приведены характеристики таблицы «Sprints»

Таблица 19 – Таблица «Sprints» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	SprintId	int

Продолжение таблицы 19

№	Название	Идентификатор	Тип данных, длина
2	Название	Name	varchar(255)
3	Описание	Description	varchar(255)
4	Идентифкатор состояния	SprintStateId	int
5	Дата начала	DateStart	datetime
6	Дата окончания	DateEnd	datetime

В таблице 20 приведены характеристики таблицы «SprintTasks».

Таблица 20 – Таблица «SprintTasks» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	SprintTaskId	int
2	Название	SprintId	int
3	Цена билета	TaskId	int

В таблице 21 приведены характеристики таблицы «Likes».

Таблица 21 – Таблица «Likes» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	LikeId	int
2	Идентификатор пользователя	LikerId	int
3	Идентификатор добавленного в избранное пользователя	LikeeId	int

В таблице 22 приведены характеристики таблицы «Messages».

Таблица 22 – Таблица «Messages» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	MessageId	int
2	Идентификатор отправителя	SenderId	int
3	Идентификатор получателя	RecipientId	int
4	Содержание сообщения	Content	varchar(255)

Продолжение таблицы 22

№	Название	Идентификатор	Тип данных, длина
5	Метка прочтения	IsRead	int
6	Дата прочтения	DateRead	datetime
7	Дата отправки	MessageSent	datetime
8	Метка удаления отправителем	SenderDeleted	int
9	Метка удаления получателем	RecipientDeleted	int

В таблице 23 приведены характеристики таблицы «SprintStates».

Таблица 23 – Таблица «SprintStates» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	SprintStateId	int
2	Название	Name	varchar(255)

В таблице 24 приведены характеристики таблицы «TaskComments».

Таблица 24 – Таблица «TaskComments» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	TaskCommentId	int
2	Идентификатор задачи	TaskId	int
3	Идентификатор автора	AuthorId	int
4	Текст комментария	Context	varchar(2048)
5	Дата	Date	datetime

В таблице 25 приведены характеристики таблицы «Photos».

Таблица 25 – Таблица «Photos» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	PhotoId	int
2	Ссылка на ресурс	Url	varchar(255)
3	Описание	Description	varchar(255)
4	Дата добавления	DateAdded	datetime
5	Метка основного фото	IsMain	bit
6	Идентификатор ресурса	PublicId	varchar(255)

Продолжение таблицы 25

№	Название	Идентификатор	Тип данных, длина
7	Идентификатор пользователя	UserId	int

Для взаимодействия приложения с базой данных будет использоваться Entity Framework, который имеет возможность использования подхода «code-first», при котором база данных генерируется автоматически на основе моделей, определенных в коде приложения.

4 Реализация системы управления

4.1 Кодирование системы управления

В основе веб-приложения используется шаблон программирования MVC. То есть модель данных, представление и бизнес-логика разделены на отдельные компоненты.

Для работы с данными, хранящимися в БД SQLite, используется кроссплатформенный фреймворк Entity Framework Core, позволяющий манипулировать данными в базе данных с использованием кода, написанного на языке C#, вместо создания запросов на языке SQL.

Серверная и клиентская части являются независимыми проектами. Рассмотрим их структуру.

Серверная часть имеет следующую структуру файлов с исходным кодом:

- Controllers – папка с классами контроллеров, содержащими основную логику приложения;
- Data – папка с классами для работы с базой данных;
- Dtos – папка с классами DTO (Data Transfer Object) для передачи и приема данных клиенту и от клиента;
- Helpers – папка с различными вспомогательными классами;
- Migrations – папка с классами, созданными Entity Framework при добавлении моделей в базу данных;
- Models – папка с моделями (сущностями), которые отображаются в базе данных;
- Properties – папка с настройками запуска приложения;
- Program.cs – точка входа приложения;
- Startup.cs – описание действий при запуске приложения.

Рассмотрим назначение файлов в приведенных выше папках:

Controllers – классы контроллеров:

- AuthController.cs – класс контроллера для запросов, связанных с аутентификацией;
- MessagesController.cs – класс контроллера для запросов, связанных с системой личных сообщений;
- PhotosController.cs – класс контроллера для запросов, связанных с фотографиями;
- UsersController.cs – класс контроллера для запросов, связанных с действиями с профилями пользователей;
- TasksControllers.cs – класс контроллера для запросов, связанных с действиями с задачами и спринтами.

Data – классы БД:

- AuthRepository.cs – класс для добавления и получения данных об аутентификации пользователей, включающий методы хеширования паролей;
- DataContext.cs – главный класс, определяющий сущности и их свойства, которые переносятся в базу данных при запуске процесса миграции;
- IAuthRepository.cs – интерфейс для класса аутентификации;
- ITfxRepository.cs – главный интерфейс, определяющий методы доступа к данным в БД;
- Seed.cs – класс для создания и заполнения базы данных при ее отсутствии (используется только при разработке);
- TfxRepository.cs – класс, содержащий методы доступа к записям в базе данных;
- UserSeedData.json – файл, данные из которого использует класс Seed для заполнения новой БД.

Dtos – классы для передачи данных:

- MessageForCreationDto.cs, MessageToReturnDto.cs – классы для передачи и приема данных о личных сообщениях;

- PhotoForCreationDto.cs, PhotoForReturnDto.cs, PhotosForDetailedDto.cs – классы для передачи и приема данных об изображениях;
- UserForDetailedDto.cs, UserForListDto.cs, UserForLoginDto.cs, UserForRegisterDto.cs, UserForUpdateDto.cs – классы для передачи и приема данных о пользователях;
- TaskForDetailedDto.cs, TaskForListDto.cs, TaskForCreationDto.cs, TaskForUpdateDto.cs – классы для передачи и приема данных о задачах;
- SprintForCreationDto.cs, SprintForDetailedDto.cs – класс для создания и просмотра спринтов.

Helpers – вспомогательные классы:

- AutoMapperProfiles.cs – класс для добавления новых возможностей для класса AutoMapper, позволяющего автоматически сопоставлять объекты разных классов;
- CloudinarySettings.cs – класс для настройки облачного сервиса Cloudinary для хранения файлов, загружаемых в систему;
- Extensions.cs – класс, содержащий методы расширения;
- LogUserActivity.cs – класс, регистрирующий обращения пользователей к API и обновляющий их время последней активности;
- MessageParams.cs – класс с параметрами для получения страницы сообщений;
- PagedList.cs – класс, позволяющий получать страницы данных вместо всех сохраненных данных;
- PaginationHeader.cs – класс, добавляющий в ответ на запрос заголовки с информацией о полученной странице, если данные получаются не целиком, а отдельными страницами;
- UserParams.cs – класс с параметрами для получения страницы пользователей;
- TaskParams.cs – класс с параметрами для получения страницы задач;

- SprintParams.cs – класс с параметрами для получения страницы спринтов.

Models – отображение данных из базы данных:

- Like.cs – информация о пользователях в избранном;
- Message.cs – информация о личных сообщениях;
- Photo.cs – информация о сохраненных в облаке изображениях;
- User.cs – информация о пользователях;
- Task.cs – информация о задачах;
- Sprint.cs – информация о спринтах.

Клиентское приложение является SPA – Single Page Application (одностраничное приложение), реализованное с помощью фреймворка Angular [29]. Особенность Angular заключается в том, что приложение состоит из изолированных компонентов, которые могут взаимодействовать друг с другом с помощью специальных конструкций. Каждый компонент состоит из трех файлов: шаблон, класс и таблица стилей. Шаблон является HTML-страницей, которая определяет структуру компонента. Класс является файлом с кодом на языке TypeScript, который определяет поведение компонента. Таблица стилей же является CSS-файлом, который применяется только внутри компонента, к которому относится.

Рассмотрим компоненты разрабатываемого приложения:

- home.component – главная страница приложения;
- lists.component – страница пользователей, добавленных в избранное;
- member-card.component – компонент карточки пользователя для отображения в списках;
- member-detail.component – компонент страницы профиля пользователей;
- member-edit.component – компонент для редактирования профиля пользователя;

- `member-list.component` – компонент для отображения всех пользователей в системе;
- `member-messages.component` – компонент для отправки и получения личных сообщений;
- `photo-editor.component` – компонент для загрузки и удаления изображений;
- `messages.component` – компонент, отображающий список полученных и отправленных сообщений;
- `nav.component` – компонент навигации, располагающийся в верхней части приложения на любом экране;
- `register.component` – компонент для создания нового пользователя;
- `task-detail.component` – компонент для создания, просмотра и редактирования задач;
- `task-list.component` – компонент, содержащий элементы для отображения списка задач;
- `task-comments.component` – компонент, содержащий комментарии к задаче и позволяющий добавлять новые комментарии;
- `sprint-detail.component` – компонент для создания и просмотра спринтов;
- `sprint-list.component` – компонент, содержащий элементы для отображения списка спринтов.

Помимо компонентов в папке с проектом SPA присутствуют следующие файлы:

- `auth.guard.ts` – запрещает навигацию неавторизованным пользователям;
- модели: `messages.ts`, `paginations.ts`, `photo.ts`, `task.ts`, `task-comment.ts`, `user.ts`, `sprint.ts` – отображают данные, хранящиеся в базе данных;

- резолверы: `list.resolver.ts`, `member-detail.resolver.ts`, `member-edit.resolver.ts`, `member-list.resolver.ts`, `messages.resolver.ts`, `task-list.resolver.ts`, `task.resolver.ts`, `sprint.resolver.ts`, `sprint-list.resolver.ts` – классы, используемые маршрутизатором Angular для вызова компонентов при переходе на определенные ссылки;
- `alertify.service.ts` – сервис, дающий удобный доступ к подключаемому компоненту Alertify для отображения уведомлений на странице;
- `auth.service.ts` – сервис, содержащий методы, связанные с аутентификацией;
- `user.service.ts` – сервис, содержащий методы для работы с данными о пользователях;
- `task.service.ts` – сервис, содержащий методы для работы с данными о задачах;
- `sprint.service.ts` – сервис, содержащий методы для работы с данными о спринтах.

На рисунке 19 представлена одна из сгенерированных страниц.

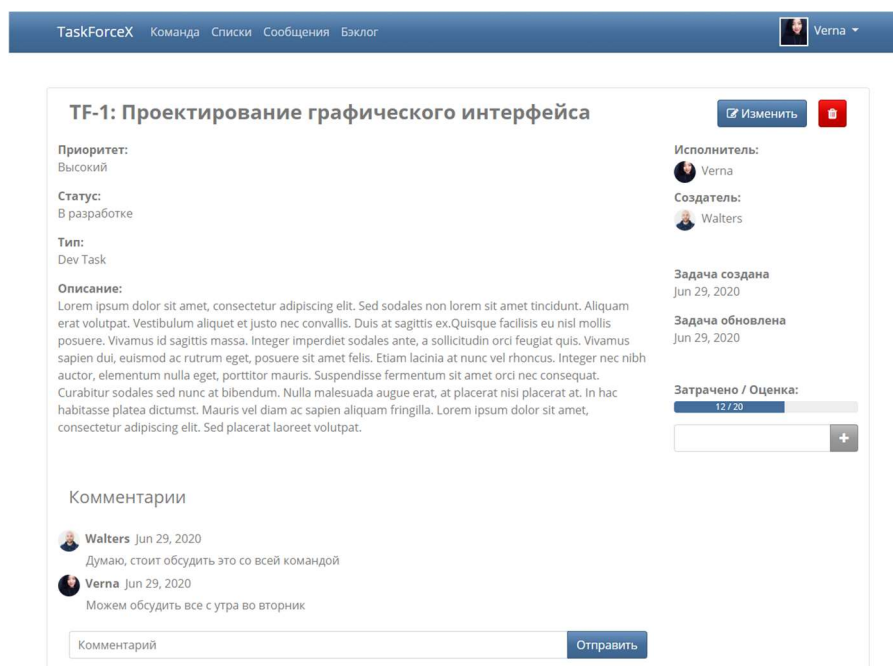


Рисунок 19 – Интерфейс страницы задачи

Таким образом в разрабатываемом приложении есть средства для генерации динамических страниц и заполнения их данными из базы данных.

4.2 Развёртывание веб-приложения

Для работы веб-приложения система бронирования необходим веб-сервер с поддержкой платформы .NET Core, а также реляционная база данных, с поддержкой языка запросов SQL. Фреймворк .NET Core позволяет использовать одну из следующих операционных систем для установки серверной части: Windows 10, Windows Server, macOS, Linux. В качестве веб-сервера рекомендуется использовать веб-сервер IIS, хотя выбор практически не ограничивается, так как при небольших изменениях можно использовать практически любой сервер. В качестве системы управления баз данных рекомендуется использовать SQLite, так как во время разработки использовалась именно эта СУБД.

В качестве клиента приложения выступает любой современный веб-браузер для любых платформ, в том числе и мобильных. В частности, страницы должны корректно отображаться в самых популярных браузерах:

- Google Chrome;
- Mozilla Firefox;
- Opera;
- Safari;
- Google Chrome для Android и iOS;
- Mozilla Firefox для Android и iOS;
- Safari для iOS;

Для установки и использования системы бронирования необходимо поместить все файлы веб-приложения в папку домена веб-сервера и настроить подключение к базе данных.

На рисунке 20 представлена диаграмма развёртывания веб-приложения.

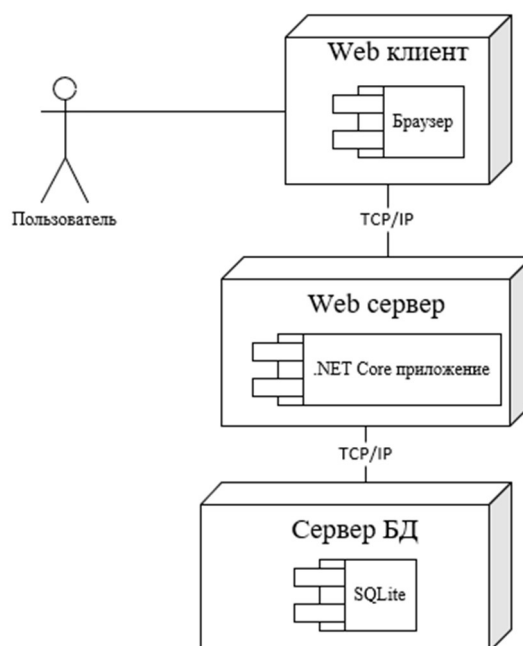


Рисунок 20 – Диаграмма развёртывания веб-приложения

На данной диаграмме развёртывания показана конфигурация обрабатывающих узлов, на которых выполняется разработанное веб-приложение и его компоненты, размещенные в этих узлах.

5 Анализ качества программного продукта

5.1 Тестирование

Для тестирования работоспособности разрабатываемого приложения будут применены следующие методики тестирования программного обеспечения: функциональное тестирование, структурное тестирование и модульное тестирование.

После анализа требований и вариантов использования были выделены самые часто возникающие сценарии использования системы управления разработкой программного обеспечения, которые необходимо покрыть тестами для обеспечения максимальной надежности работы и минимизации количества ошибок на этапе кодирования:

- 1) загрузка страницы;
- 2) загрузка страницы ошибки при обращении к несуществующему ресурсу;
- 3) настройка нового пользователя;
- 4) авторизация;
- 5) поиск задачи;
- 6) просмотр сведений о задаче;
- 7) редактирование задачи;
- 8) удаление задачи;
- 9) создание задачи;
- 10) создание спринта;
- 11) редактирование профиля;
- 12) просмотр профиля другого участника команды.

На основе приведенных выше сценариев был составлен план тестирования и проведено функциональное тестирование. Результат тестирования приведен в таблице 26.

Таблица 26 – План тестирования

№	Названия тестового случая	Тест	Результат
1	Загрузка страницы	Будет отображена запрашиваемая страница	Страница успешно загружена
2	Загрузка страницы ошибки при обращении к несуществующему ресурсу	Будет отображена страница 404	Страница 404 успешно загружена
3	Настройка нового пользователя	При первом входе пользователю будет предложено создать новый пароль	Новый пароль запрошен и успешно сохранен
4	Авторизация	При вводе корректных данных для авторизации пользователь будет перенаправлен на главную страницу	Пользователь попадает на главную страницу и получает доступ к системе
5	Поиск задачи	При вводе данных в строку поиска система отображает подходящие задачи	Отображен список задач, соответствующих условиям поиска
6	Просмотр сведений о задаче	При запросе страницы задачи отображается информация о запрашиваемой задаче	Корректно отображена страница требуемой задачи
7	Редактирование задачи	При вводе новых данных задача обновится	После сохранения изменений данные о задаче обновлены
8	Удаление задачи	После удаления задача не должна отображаться в списке задач	После удаления задача не отображается в списке всех задач и не появляется в результатах поиска
9	Создание задачи	При вводе и сохранении данных о новой задаче она отображается в бэклоге	Новая задача создана и отображается в бэклоге
10	Создание спринта	После ввода данных и добавления задач в новый спринт он отображается в списке	Новый спринт создан и корректно отображает список задач

Продолжение таблицы 26

№	Названия тестового случая	Тест	Результат
11	Редактирование профиля	При вводе новых данных профиль обновится	После сохранения изменений данные профиля обновлены
12	Просмотр профиля другого участника команды	Должны быть отображены данные запрашиваемого профиля	Корректно отображены запрашиваемые данные

Для авторизации было проведено тестирование методом выделения классов эквивалентности. Результаты тестирования приведены в таблице 27.

Таблица 27 – Результаты тестирования

№	Входное условие	Пример	Правильность
1	Корректный адрес почты	example@mail.com	+
2	Некорректный адрес почты	examplemailcom	-
3	Пустая строка в поле для ввода почты		-
4	Корректный пароль	CorrectPass120	+
5	Пароль короче 4 символов	pass	-
6	Пароль длиннее 8 символов	nvhfYurhFJu57svcbxsA574636373728Tr	-
7	Пустая строка в поле для ввода пароля		-

Сообщение об ошибке представлено на рисунке 21.

The screenshot shows the TaskForceX application interface. At the top, there is a dark blue header with the 'TaskForceX' logo on the left, a search bar containing the text 'test' in the center, and a green 'Войти' (Login) button on the right. Below the header, the main content area has a white background. It features the title 'Весь Scrum в одном приложении' (All Scrum in one application) in a large, bold, dark font. Underneath the title is the subtitle 'Управление разработкой программных продуктов по методологии Scrum' (Management of software product development by the Scrum methodology) in a smaller, regular font. Below the subtitle are two buttons: a dark blue 'Регистрация' (Registration) button and a light blue 'Что такое Scrum?' (What is Scrum?) button. In the bottom right corner of the main content area, there is a red rectangular box containing the text 'Неверное имя/пароль' (Incorrect name/password) in white, indicating an authentication error.

Рисунок 21 – Сообщение о некорректно введенных данных

Так как веб-приложение состоит из клиентской части и серверной части, то и тестирование каждой части нужно проводить, учитывая особенности каждой технологии. Серверная часть разрабатывается на языке C# с использованием фреймворка для веб-приложений ASP.NET Core и фреймворка для работы с базами данных Entity Framework. Для создания клиентской части используется TypeScript фреймворк Angular.

Рассмотрим подробнее инструменты тестирования, которые будут применяться.

Для тестирования серверной части приложения применяются средства тестирования, встроенные в интегрированную среду разработки Visual Studio, и библиотека Mock для проведения тестирования отдельных компонентов без учета влияния внешних сущностей. Методы API (application programming interface) тестируются с использованием специального ПО для работы с

API – Postman, которое позволяет сохранять коллекции запросов к серверу и проводить автоматизированное тестирование.

Клиентскую часть приложения, написанную на TypeScript с использованием фреймворка Angular, эффективнее и быстрее тестировать встроенными средствами Angular.

На основе функциональных требований были выделены следующие основные модули для тестирования:

- подсистема авторизации;
- модуль работы с задачами;
- модуль работы со спринтами;

На основе вариантов использования для модуля работы с задачами был составлен список тестовых случаев:

- поиск задач;
- просмотр данных о задаче;
- изменение данных о задаче;
- создание задачи;
- удаление задачи;

Используя метод покрытия условий, был составлен список тестов для вариантов использования этого модуля, представленного классом TaskService. Список тестов представлен в таблице 28.

Таблица 28 – Тесты подсистемы управления задачами

№	Метод класса	Тест	Результат	Тест пройден
1	GetAll	Получение всех задач	Возвращен список всех задач	+
2	GetAllByName	Получение задач по заданному имени	Возвращен список подходящих задач	+
3	GetTask	Получение конкретной задачи	Возвращена требуемая задача	+
4	UpdateTask	Обновление данных о задаче	Данные обновлены	+
5	UpdateTask	Обновление данных удаленной задачи	Данные не обновлены	+
6	UpdateTask	Обновление несуществующей задачи	Сообщение об ошибке	+

Продолжение таблицы 28

№	Метод класса	Тест	Результат	Тест пройден
7	UpdateTask	Обновление задачи с некорректными параметрами	Сообщение об ошибке	+
8	CreateTask	Создание задачи	Сообщение об успешном создании задачи	+
9	DeleteTask	Удаление задачи	Сообщение об успешном удалении	+
10	DeleteTask	Удаление несуществующей задачи	Сообщение об ошибке	+

На основе составленных тестов были созданы юнит-тесты на языке программирования C#. В приложении Л представлен исходный код тестов.

После было запущено автоматизированное модульное тестирование, результаты которого представлены на рисунке 22.

Test	Duration	Traits	Error Message
TaskForce.Tests (10)	858 ms		
TaskForce.Tests.TaskService (10)	858 ms		
TaskServiceTests (10)	858 ms		
Tasks_CreateTask	828 ms		
Tasks_DeleteTask_Correct	4 ms		
Tasks_DeleteTask_Deleted	< 1 ms		
Tasks_GetAll	6 ms		
Tasks_GetAllByName	2 ms		
Tasks_GetTask	4 ms		
Tasks_UpdateTask_Correct	2 ms		
Tasks_UpdateTask_Deleted	10 ms		
Tasks_UpdateTask_IncorrectPara...	2 ms		
Tasks_UpdateTask_NonExisting	< 1 ms		

Рисунок 22 – Результат выполнения юнит-тестов

Таким образом все методы класса оказались покрыты тестами. Все функциональные требования выполнены, что подтверждается отсутствием ошибок после проведения тестов.

Юнит-тесты других классов аналогичны приведенному выше.

Тестирование клиентской части полностью проводится вручную, поэтому использование встроенных средств тестирования Angular не требуется.

5.2 Метрики кода

В качестве основной метрики используется Maintainability Index – комплексный показатель качества кода, который автоматически рассчитывается средствами Visual Studio. Данная метрика зависит от других метрик, среди которых: объем программы (Halstead Volume, HV), цикломатическая сложность (Cyclomatic Complexity, CC) и количество строк кода (Lines of Code, LOC). Вычисляется данная метрика по формуле 1.

$$MI = \text{MAX}(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * 100 / 171) \quad (1)$$

Данный показатель принимает значения от 0 до 100, чем выше значение – тем более простой считается поддержка кода. Хорошим показателем считается значение от 20 до 100 (зеленая метка), приемлемым – от 10 до 20 (желтая метка), плохим – менее 10 (красная метка).

После применения автоматизированного программного средства, встроенного в Visual Studio, для анализа и расчёта метрик были получены следующие показатели, представленные в таблице 29.

Таблица 29 – Метрики кода для разработанного веб-приложения

Метрика	Значение
Размерно-ориентированные метрики	
Количество строк кода	4283
Логических строк кода	2975
Количество строк комментариев	212
Объектно-ориентированные метрики	
Количество классов	51
Количество методов	136
Сложность кода	
Средняя цикломатическая сложность по классу	12.2

А также получен показатель качества кода для каждого класса. Показатели Maintainability Index приведены в таблице 30.

Таблица 30 – Показатель качества кода основных компонентов

№	Компонент	MI
1	TaskForceX.Api.Program	69
2	TaskForceX.Api.Startup	65
3	TaskForceX.Api.Controllers.AuthController	63
4	TaskForceX.Api.Controllers.MessagesController	64
5	TaskForceX.Api.Controllers.PhotosController	58
6	TaskForceX.Api.Controllers.UsersController	66
7	TaskForceX.Api.Controllers.TasksController	67
8	TaskForceX.Api.Controllers.SprintsController	63
9	TaskForceX.Api.Data.AuthRepository	72
10	TaskForceX.Api.Data.DataContext	81
11	TaskForceX.Api.Data.IAuthRepository	100
12	TaskForceX.Api.Data.ITfxRepository	100
13	TaskForceX.Api.Data.TfxRepository	67
14	TaskForceX.Api.Data.Seed	67
15	TaskForceX.Api.Dtos.MessageForCreationDto	100
16	TaskForceX.Api.Dtos.MessageToReturnDto	100
17	TaskForceX.Api.Dtos.PhotoForCreationDto	100
18	TaskForceX.Api.Dtos.PhotoForReturnDto	100
19	TaskForceX.Api.Dtos.PhotosForDetailedDto	100
20	TaskForceX.Api.Dtos.UserForDetailedDto	100
21	TaskForceX.Api.Dtos.UserForListDto	100
22	TaskForceX.Api.Dtos.UserForLoginDto	100
23	TaskForceX.Api.Dtos.UserForRegisterDto	93
24	TaskForceX.Api.Dtos.UserForUpdateDto	100
25	TaskForceX.Api.Dtos.TaskForDetailedDto	100
26	TaskForceX.Api.Dtos.TaskForListDto	100
27	TaskForceX.Api.Dtos.SprintForCreationDto	100
28	TaskForceX.Api.Dtos.SprintForListDto	100
29	TaskForceX.Api.Helpers.AutoMapperProfiles	48
30	TaskForceX.Api.Helpers.CloudinarySettings	10
31	TaskForceX.Api.Helpers.Extensions	75
32	TaskForceX.Api.Helpers.LogUserActivity	67
33	TaskForceX.Api.Helpers.PagedList<T>	89
34	TaskForceX.Api.Helpers.PaginationHeader	93
	Средний показатель качества кода проекта	81,68

Подводя итог, можно сказать, что проект имеет высокий показатель качества кода.

Заключение

В результате проведенного исследования были изучены особенности принципы работы и способы разработки веб-приложений. Было проведено исследования вариантов использования системы для организации управления разработкой программных проектов по методологии SCRUM.

Была рассмотрена и проанализирована актуальная техническая литература по проектированию и разработке информационных систем для сети Интернет.

Проведя анализ существующих сервисов и систем управления разработкой программных проектов по методологии Scrum, было установлено, что такие системы или не отвечают всем требованиям методологии, или слишком сложны и дороги в использовании.

Для того, чтобы учесть все аспекты работы с приложением, были исследованы варианты использования системы для организации управления разработкой программных продуктов, составлены диаграммы прецедентов и последовательности действий для некоторых вариантов использования. Была определена подходящая структура приложения, а также проанализированы средства разработки веб-приложений.

Было проведено проектирование программных средства, структур хранения данных и пользовательского интерфейса. Разработаны макеты интерфейса клиентской части приложения в виде веб-страницы. Был определен и рассмотрен процесс обработки данных, а также составлена диаграмма компонентов. Была спроектирована структура хранения данных и составлена диаграмма базы данных.

После проектирования было реализовано веб-приложение для управления разработкой программных продуктов по методологии Scrum. Приложение имеет удобный пользовательский интерфейс и позволяет пользователям добавлять задачи в бэклог и планировать спринты, а также связываться с другими пользователями путем личных сообщений и добавлять пользователей в избранное для быстрого доступа. При этом функционал

приложения доступен только для пользователей, прошедших аутентификацию.

Для проверки качества были применены некоторые методологии тестирования: функциональное, структурное и модульное тестирование. Главные классы приложения покрыты модульными тестами и показывают отсутствие ошибок. Таким образом, можно сделать вывод о том, что приложение работает без ошибок и соответствует заявленным требованиям.

Было проведено планирование проекта с выделением длительности работ и взаимосвязей между ними. Составлен план стадий и этапов проектирования и разработки веб-приложения, и на его основе была построена диаграмма Ганта. Анализ этапов и их продолжительности показал, что на проектирование и разработку системы понадобится 376 часов. Также на основе планирования был произведен расчет стоимости всего проекта. Были учтены расходы на амортизацию, заработную плату, социальные выплаты и прочие материальные расходы. Была составлена диаграмма структуры расходов. В результате расчета стоимость проекта составила 253 542,47 рублей.

Подводя итог, можно сказать, что поставленная цель работы выполнены.

Список использованных источников

- 1 Scrum.org [Электронный ресурс]. URL: <https://www.scrum.org/resources/what-is-scrum> (дата обращения: 23.04.2020).
- 2 Икуджиро, Н. Компания - создатель знания. Зарождение и развитие инноваций в японских фирмах / Н. Икуджиро, Т.Хиротака – М.: Озон, 2011. - с. 52.
- 3 Сазерленд, Д. Scrum. Революционный метод управления проектами / Д. Сазерленд – М.: Миф, 2004. - с. 41.
- 4 Швабер, К. Scrum – Гибкое управление продуктом и бизнесом / К. Швабер – М: Альпина Паблишер, 2019. – с.22-141.
- 5 Trello.com [Электронный ресурс]. URL: <https://trello.com/> (дата обращения 23.04.2020).
- 6 Андерсон, Д. Канбан. Альтернативный путь в Agile / Д. Андерсон – М.: МИФ, 2019. – с.44
- 7 Стеарт, П. Восходящий Канбан. Краткое руководство / П. Стеарт – М.: Альпина Паблишер, 2016. – с.15.
- 8 Planview.com [Электронный ресурс] URL: <https://www.planview.com/products-solutions/products/leankit/>. (дата обращения: 23.04.2020).
- 9 Atlassian.com [Электронный ресурс] URL: <https://www.atlassian.com/software/jira>. (дата обращения: 22.04.2020).
- 10 Product Feature – nTask [Электронный ресурс] URL: <https://www.ntaskmanager.com/product/>. (дата обращения: 22.04.2020).
- 11 About Asana [Электронный ресурс] URL: <https://asana.com/product>. (дата обращения: 23.04.2020).
- 12 Маркот, И. Отзывчивый веб-дизайн / И. Маркот – М.: МИФ, 2012. – с.20-129.
- 13 Can I use [Электронный ресурс] URL: <https://caniuse.com/>. (дата обращения: 23.04.2020).
- 14 Голдштейн, И. Scrum без ошибок / И. Голдштейн – М.:МИФ, 2012. – с.2-34.

- 15 Буч, Г. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. / Г. Буч, Д. Рамбо, И. Якобсон. – М.: ДМК Пресс, 2006. – 496 с.
- 16 Фримен, А. Angular для профессионалов / А. Фримен – СПб.: Питер, 2018. 800 с.
- 17 Пьюривал, С. Основы разработки веб-приложений / С. Пьюривал. – СПб.: Питер, 2015. - 272 с.
- 18 Спецификация HTML5. [Электронный ресурс] URL: <https://www.w3.org/TR/html/> (дата обращения: 19.05. 2020).
- 19 Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. – М: Эксмо, 2013. – 480 с.
- 20 Фримен, А. ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов / А. Фримен М.:Вильямс, 2018. – 736 с.
- 21 Сандерсон, С. ASP .NET MVC Framework с примерами на C# / С. Сандерсон М.:Вильямс, 2010. – 560 с.
- 22 PHP. Руководство по PHP. [Электронный ресурс] URL: <http://php.net/manual/ru/> . (дата обращения: 19.04.2020).
- 23 Денис Колисниченко. Движок для вашего сайта. CMS Joomla!, Slaed, PHP-Nuke. — Петербург: БХВ, 2008. — 352 с.
- 24 Уолл, Л. Программирование на Perl / Л. Уолл, Т. Кристиансен, Д. Орвант – М.:Символ-Плюс, 2006. – с. 23-25
- 25 TIOBE [Электронный ресурс] URL: <https://www.tiobe.com/tiobe-index/perl/>. (дата обращения: 15.04.2020).
- 26 Don't repeat yourself – Wikipedia [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Don't_repeat_yourself. (дата обращения: 15.04.2020).
- 27 Convention over configuration – Wikipedia [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Convention_over_configuration. (дата обращения: 15.04.2020).

Приложение А
Листинг кода файла Program.cs
(рекомендуемое)

```

using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.DependencyInjection;
using TaskForceX.API.Data;

namespace TaskForceX.API
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = CreateHostBuilder(args).Build();
            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
                    var context =
services.GetRequiredService<DataContext>();
                    context.Database.Migrate();
                    Seed.SeedUsers(context);
                }
                catch (Exception ex)
                {
                    var logger =
services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred during
migration");
                }
            }
            host.Run();
        }

        public static IHostBuilder CreateHostBuilder(string[]
args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                })
            ;
    }
}

```

Приложение Б
Листинг кода файла Startup.cs
(рекомендуемое)

```

using System.Text;
using System.Net;
using AutoMapper;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Diagnostics;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using TaskForceX.API.Data;
using TaskForceX.API.Helpers;

namespace TaskForceX.API
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this
        method to add services to the container.
        public void ConfigureServices(IServiceCollection
        services)
        {
            services.AddDbContext<DataContext>(x =>
            x.UseSqlite(Configuration.GetConnectionString("DefaultConnection
            ")));
            services.AddControllers().AddNewtonsoftJson(opt =>
            {
                opt.SerializerSettings.ReferenceLoopHandling =
                Newtonsoft.Json.ReferenceLoopHandling.Ignore;
            });
            services.AddCors();

            services.Configure<CloudinarySettings>(Configuration.GetSection(
            "CloudinarySettings"));

            services.AddAutoMapper(typeof(TfxRepository).Assembly);
            services.AddScoped<IAuthRepository,
            AuthRepository>();
            services.AddScoped<ITfxRepository, TfxRepository>();

            services.AddAuthentication(JwtBearerDefaults.AuthenticationSchem
            e)
                .AddJwtBearer(options => {
                    options.TokenValidationParameters = new
                    TokenValidationParameters

```

```

        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.ASCII
.GetBytes(Configuration.GetSection("AppSettings:Token").Value)),
            ValidateIssuer = false,
            ValidateAudience = false
        };
    });
    services.AddScoped<LogUserActivity>();
}

// This method gets called by the runtime. Use this
method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler(builder => {
            builder.Run(async context => {
                context.Response.StatusCode =
(int)HttpStatusCode.InternalServerError;
                var error =
context.Features.Get<IExceptionHandlerFeature>();
                if (error != null)
                {
                    context.Response.AddApplicationError(error.Error.Message);
                    await
context.Response.WriteAsync(error.Error.Message);
                }
            });
        });
    }
    // app.UseHttpsRedirection();
    app.UseRouting();
    app.UseCors(x =>
x.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
    app.UseAuthentication();
    app.UseAuthorization();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
}
}

```

Приложение В
Листинг кода файла Extensions.cs
(рекомендуемое)

```

using System;
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;

namespace TaskForceX.API.Helpers
{
    public static class Extensions
    {
        public static void AddApplicationError(this HttpResponseMessage
response, string message)
        {
            response.Headers.Add("Application-Error", message);
            response.Headers.Add("Access-Control-Expose-
Headers", "Application-Error");
            response.Headers.Add("Access-Control-Allow-Origin",
"*");
        }

        public static int CalculateAge(this DateTime dt)
        {
            var age = DateTime.Today.Year - dt.Year;
            if (dt.AddYears(age) > DateTime.Today) age--;
            return age;
        }

        public static void AddPagination(this HttpResponseMessage
response,
            int currentPage, int itemsPerPage, int totalItems,
int totalPages)
        {
            var paginationHeader = new
PaginationHeader(currentPage, itemsPerPage, totalItems,
totalPages);
            var camelCaseFormatter = new
JsonSerializerSettings();
            camelCaseFormatter.ContractResolver = new
CamelCasePropertyNamesContractResolver();
            response.Headers.Add("Pagination",
                JsonConvert.SerializeObject(paginationHeader,
camelCaseFormatter));
            response.Headers.Add("Access-Control-Expose-
Headers", "Pagination");
        }
    }
}

```

Приложение Г
Листинг кода файла AuthRepository.cs
(рекомендуемое)

```

using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using TaskForceX.API.Models;

namespace TaskForceX.API.Data
{
    public class AuthRepository : IAuthRepository
    {
        private readonly DataContext _context;

        public AuthRepository(DataContext context)
        {
            this._context = context;
        }

        public async Task<User> Register(User user, string
password)
        {
            byte[] passwordHash, passwordSalt;
            CreatePasswordHash(password, out passwordHash, out
passwordSalt);

            user.PasswordHash = passwordHash;
            user.PasswordSalt = passwordSalt;

            await _context.Users.AddAsync(user);
            await _context.SaveChangesAsync();

            return user;
        }

        private void CreatePasswordHash(string password, out
byte[] passwordHash, out byte[] passwordSalt)
        {
            using (var hmac = new
System.Security.Cryptography.HMACSHA512())
            {
                passwordSalt = hmac.Key;
                passwordHash =
hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));
            }
        }

        public async Task<User> Login(string username, string
password)
        {
            var user = await _context.Users.Include(p =>
p.Photos).FirstOrDefaultAsync(x => x.Username == username);

            if (user == null)
                return null;

            if (!VerifyPasswordHash(password, user.PasswordHash,
user.PasswordSalt))
                return null;
        }
    }
}

```

```

        return user;
    }

    private bool VerifyPasswordHash(string password, byte[]
passwordHash, byte[] passwordSalt)
    {
        using (var hmac = new
System.Security.Cryptography.HMACSHA512(passwordSalt))
        {
            var computedHash =
hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));
            for (int i = 0; i < computedHash.Length; i++)
            {
                if (computedHash[i] != passwordHash[i])
return false;
            }
        }
        return true;
    }

    public async Task<bool> UserExists(string username)
    {
        if (await _context.Users.AnyAsync(x => x.Username ==
username))
            return true;

        return false;
    }
}

```

Приложение Д
Листинг файла UsersController.cs
(рекомендуемое)

```

using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using System.Security.Claims;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authorization;
using TaskForceX.API.Data;
using TaskForceX.API.Dtos;
using TaskForceX.API.Helpers;
using TaskForceX.API.Models;
using AutoMapper;

namespace TaskForceX.API.Controllers
{
    [ServiceFilter(typeof(LogUserActivity))]
    [Authorize]
    [Route("api/[controller]")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        private readonly ITfxRepository _repo;
        private readonly IMapper _mapper;

        public UsersController(ITfxRepository repo, IMapper
mapper)
        {
            this._repo = repo;
            this._mapper = mapper;
        }

        [HttpGet]
        public async Task<IActionResult>
GetUsers([FromQuery]UserParams userParams)
        {
            var currentUserId =
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value);
            var userFromRepo = await
_repo.GetUser(currentUserId);
            userParams.UserId = currentUserId;
            if (string.IsNullOrEmpty(userParams.Gender))
            {
                userParams.Gender = userFromRepo.Gender ==
"male" ? "female" : "male";
            }
            var users = await _repo.GetUsers(userParams);
            var usersToReturn =
_mapper.Map<IEnumerable<UserForListDto>>(users);
            Response.AddPagination(users.CurrentPage,
users.PageSize,
                users.TotalCount, users.TotalPages);
            return Ok(usersToReturn);
        }

        [HttpGet("{id}", Name = "GetUser")]
        public async Task<IActionResult> GetUser(int id)

```

```

        {
            var user = await _repo.GetUser(id);
            var userToReturn =
_mapper.Map<UserForDetailedDto>(user);
            return Ok(userToReturn);
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> UpdateUser(int id,
UserForUpdatedDto userForUpdatedDto)
        {
            if (id !=
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))
                return Unauthorized();

            var userFromRepo = await _repo.GetUser(id);
            _mapper.Map(userForUpdatedDto, userFromRepo);

            if (await _repo.SaveAll())
                return NoContent();

            throw new Exception($"Не удалось обновить данные о
пользователе {id}");
        }

        [HttpPost("{id}/like/{recipientId}")]
        public async Task<IActionResult> LikeUser(int id, int
recipientId)
        {
            if (id !=
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value))
                return Unauthorized();

            var like = await _repo.GetLike(id, recipientId);
            if (like != null)
                return BadRequest("Вы уже оценили этого
пользователя");
            if (await _repo.GetUser(recipientId) == null)
                return NotFound();

            like = new Like
            {
                LikerId = id,
                LikeeId = recipientId
            };

            _repo.Add<Like>(like);
            if (await _repo.SaveAll())
                return Ok();

            return BadRequest("Не удалось оценить");
        }
    }
}

```

Приложение Е
Листинг кода файла `user.service.ts`
(рекомендуемое)

```

import { Injectable } from '@angular/core';
import { environment } from 'src/environments/environment';
import { HttpClient, HttpHeaders, HttpParams } from
 '@angular/common/http';
import { Observable } from 'rxjs';
import { User } from 'src/app/_models/user';
import { PaginatedResult } from '../_models/pagination';
import { map } from 'rxjs/operators';
import { Message } from '../_models/message';
import { Task } from '../_models/task';

@Injectable({
  providedIn: 'root'
})
export class UserService {
  baseUrl = environment.apiUrl;

  constructor(private http: HttpClient) { }

  getUsers(page?, itemsPerPage?, userParams?, likesParam?):
  Observable<PaginatedResult<User[]>> {
    const paginatedResult: PaginatedResult<User[]> = new
  PaginatedResult<User[]>();
    let params = new HttpParams();
    if (page != null && itemsPerPage != null) {
      params = params.append('pageNumber', page);
      params = params.append('pageSize', itemsPerPage);
    }
    if (userParams != null) {
      params = params.append('minAge', userParams.minAge);
      params = params.append('maxAge', userParams.maxAge);
      params = params.append('gender', userParams.gender);
      params = params.append('orderBy', userParams.orderBy);
    }
    if (likesParam === 'Likers') {
      params = params.append('likers', 'true');
    }
    if (likesParam === 'Likees') {
      params = params.append('likees', 'true');
    }

    return this.http.get<User[]>(this.baseUrl + 'users',
    {observe: 'response', params})
      .pipe(
        map(response => {
          paginatedResult.result = response.body;
          if (response.headers.get('Pagination') != null) {
            paginatedResult.pagination =
            JSON.parse(response.headers.get('Pagination'));
          }
          return paginatedResult;
        })
      );
  }
}

```

```

getUser(id): Observable<User> {
    return this.http.get<User>(this.baseUrl + 'users/' + id);
}

updateUser(id: number, user: User) {
    return this.http.put(this.baseUrl + 'users/' + id, user);
}

setMainPhoto(userId: number, id: number) {
    return this.http.post(this.baseUrl + 'users/' + userId +
'/photos/' + id + '/setMain', {});
}

deletePhoto(userId: number, id: number) {
    return this.http.delete(this.baseUrl + 'users/' + userId +
'/photos/' + id);
}

sendLike(id: number, recipientId: number) {
    return this.http.post(this.baseUrl + 'users/' + id +
'/like/' + recipientId, {});
}

getMessages(id: number, page?, itemsPerPage?,
messageContainer?) {
    const paginatedResult: PaginatedResult<Message[]> = new
PaginatedResult<Message[]>();
    let params = new HttpParams();
    params = params.append('MessageContainer',
messageContainer);

    if (page != null && itemsPerPage != null) {
        params = params.append('pageNumber', page);
        params = params.append('pageSize', itemsPerPage);
    }

    return this.http.get<Message[]>(this.baseUrl + 'users/' + id
+ '/messages',
    { observe: 'response', params })
    .pipe(
        map(response => {
            paginatedResult.result = response.body;
            if (response.headers.get('Pagination') != null) {
                paginatedResult.pagination =
JSON.parse(response.headers.get('Pagination'));
            }

            return paginatedResult;
        })
    );
}

getMessageThread(id: number, recipientId: number) {
    return this.http.get<Message[]>(this.baseUrl + 'users/' + id
+ '/messages/thread/' + recipientId);
}

```

```

    }

    sendMessage(id: number, message: Message) {
        return this.http.post(this.baseUrl + 'users/' + id +
'/messages', message);
    }

    deleteMessage(id: number, userId: number) {
        return this.http.post(this.baseUrl + 'users/' + userId +
'/messages/' + id, {});
    }

    markAsRead(userId: number, messageId: number) {
        this.http.post(this.baseUrl + 'users/' + userId +
'/messages/' + messageId + '/read', {})
        .subscribe();
    }

    getTasks(page?, itemsPerPage?):
Observable<PaginatedResult<Task[]>> {
    const paginatedResult: PaginatedResult<Task[]> = new
PaginatedResult<Task[]>();
    let params = new HttpParams();
    if (page != null && itemsPerPage != null) {
        params = params.append('pageNumber', page);
        params = params.append('pageSize', itemsPerPage);
    }

    return this.http.get<Task[]>(this.baseUrl + 'tasks',
    { observe: 'response', params })
        .pipe(
            map(response => {
                paginatedResult.result = response.body;
                if (response.headers.get('Pagination') != null) {
                    paginatedResult.pagination =
JSON.parse(response.headers.get('Pagination'));
                }
                return paginatedResult;
            })
        );
    }
}

```

Приложение Ж

Листинг кода файла `messages.component.ts`

(рекомендуемое)

```

import { Component, OnInit } from '@angular/core';
import { Message } from '../_models/message';
import { Pagination, PaginatedResult } from
'../_models/pagination';
import { UserService } from '../_services/user.service';
import { AuthService } from '../_services/auth.service';
import { ActivatedRoute } from '@angular/router';
import { AlertifyService } from '../_services/alertify.service';

@Component({
  selector: 'app-messages',
  templateUrl: './messages.component.html',
  styleUrls: ['./messages.component.css']
})
export class MessagesComponent implements OnInit {
  messages: Message[];
  pagination: Pagination;
  messageContainer = 'Unread';

  constructor(
    private userService: UserService,
    private authService: AuthService,
    private route: ActivatedRoute,
    private alertify: AlertifyService
  ) { }

  ngOnInit() {
    this.route.data.subscribe(
      (data) => {
        this.messages = data['messages'].result;
        this.pagination = data['messages'].pagination;
      }
    );
  }

  loadMessages() {

    this.userService.getMessages(this.authService.decodedToken.nameid,
      this.pagination.currentPage, this.pagination.itemsPerPage,
      this.messageContainer)
      .subscribe(
        (res: PaginatedResult<Message[]>) => {
          this.messages = res.result;
          this.pagination = res.pagination;
        },
        (error) => {
          this.alertify.error(error);
        }
      );
  }

  deleteMessage(id: number) {
    this.alertify.confirm('Удаление', 'Вы уверены, что хотите
    удалить сообщение?', () => {

```

```

        this.userService.deleteMessage(id,
this.authService.decodedToken.nameid)
        .subscribe(
            () => {
                this.messages.splice(this.messages.findIndex(m =>
m.id === id), 1);
                this.alertify.success('Сообщение удалено');
            },
            (error) => {
                this.alertify.error('Не удалось удалить сообщение');
            }
        );
    });
}

pageChanged(event: any): void {
    this.pagination.currentPage = event.page;
    this.loadMessages();
}
}

```

Приложение И
Листинг кода маршрутизатора routes.ts
(рекомендуемое)

```

import { Routes } from '@angular/router';
import { HomeComponent } from '../home/home.component';
import { MemberListComponent } from '../members/member-list/member-list.component';
import { MessagesComponent } from
'../messages/messages.component';
import { ListsComponent } from '../lists/lists.component';
import { AuthGuard } from '../_guards/auth.guard';
import { MemberDetailComponent } from '../members/member-detail/member-detail.component';
import { MemberDetailResolver } from '../_resolvers/member-detail.resolver';
import { MemberListResolver } from '../_resolvers/member-list.resolver';
import { MemberEditComponent } from '../members/member-edit/member-edit.component';
import { MemberEditResolver } from '../_resolvers/member-edit.resolver';
import { PreventUnsavedChanges } from '../_guards/prevent-unsaved-changes.guard';
import { ListResolver } from '../_resolvers/lists.resolver';
import { MessagesResolver } from
'../_resolvers/messages.resolver';
import { TaskListComponent } from '../tasks/task-list/task-list.component';
import { TaskListResolver } from '../_resolvers/task-list.resolver';

export const appRoutes: Routes = [
  { path: '', component: HomeComponent },
  {
    path: '',
    runGuardsAndResolvers: 'always',
    canActivate: [AuthGuard],
    children: [
      { path: 'members', component: MemberListComponent,
        resolve: { users: MemberListResolver } },
      { path: 'members/:id', component:
MemberDetailComponent,
        resolve: { user: MemberDetailResolver } },
      { path: 'member/edit', component:
MemberEditComponent,
        resolve: { user: MemberEditResolver },
canDeactivate: [PreventUnsavedChanges] },
      { path: 'messages', component: MessagesComponent,
        resolve: { messages: MessagesResolver } },
      { path: 'lists', component: ListsComponent,
        resolve: { users: ListResolver } },
      { path: 'tasks', component: TaskListComponent,
        resolve: { tasks: TaskListResolver } }
    ]
  },
  { path: '**', redirectTo: '', pathMatch: 'full' }
];

```

Приложение К

Листинг кода главного модуля `app.module.ts`

(рекомендуемое)

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule, ReactiveFormsModule } from
 '@angular/forms';
import { JwtModule } from '@auth0/angular-jwt';
import { BsDropdownModule } from 'ngx-bootstrap/dropdown';
import { TabsModule } from 'ngx-bootstrap/tabs';
import { BsDatepickerModule } from 'ngx-bootstrap/datepicker';
import { PaginationModule } from 'ngx-bootstrap/pagination';
import { ButtonsModule } from 'ngx-bootstrap/buttons';
import { ProgressbarModule } from 'ngx-bootstrap/progressbar';
import { FileUploadModule } from 'ng2-file-upload';
import { TimeagoModule } from 'ngx-timeago';

import { AppComponent } from './app.component';
import { NavComponent } from './nav/nav.component';
import { AuthService } from './_services/auth.service';
import { HomeComponent } from './home/home.component';
import { RegisterComponent } from
 './register/register.component';
import { ErrorInterceptorProvider } from
 './_services/error.interceptor';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { MemberListComponent } from './members/member-
list/member-list.component';
import { ListsComponent } from './lists/lists.component';
import { MessagesComponent } from
 './messages/messages.component';
import { RouterModule } from '@angular/router';
import { appRoutes } from './routes';
import { MemberCardComponent } from './members/member-
card/member-card.component';
import { MemberDetailComponent } from './members/member-
detail/member-detail.component';
import { AlertifyService } from './_services/alertify.service';
import { AuthGuard } from './_guards/auth.guard';
import { UserService } from './_services/user.service';
import { MemberDetailResolver } from './_resolvers/member-
detail.resolver';
import { MemberListResolver } from './_resolvers/member-
list.resolver';
import { NgxGalleryModule } from '@kolkov/ngx-gallery';
import { MemberEditComponent } from './members/member-
edit/member-edit.component';
import { MemberEditResolver } from './_resolvers/member-
edit.resolver';
import { PreventUnsavedChanges } from './_guards/prevent-
unsaved-changes.guard';
import { PhotoEditorComponent } from './members/photo-
editor/photo-editor.component';
import { ListResolver } from './_resolvers/lists.resolver';
import { MessagesResolver } from
 './_resolvers/messages.resolver';

```

```

import { MemberMessagesComponent } from './members/member-
messages/member-messages.component';
import { TaskListResolver } from './_resolvers/task-
list.resolver';
import { TaskListComponent } from './tasks/task-list/task-
list.component';
import { TaskDetailComponent } from './tasks/task-detail/task-
detail.component';
import { TaskCommentsComponent } from './tasks/task-
comments/task-comments.component';

export function tokenGetter() {
  return localStorage.getItem('token');
}

@NgModule({
  declarations: [
    AppComponent,
    NavComponent,
    HomeComponent,
    RegisterComponent,
    MemberListComponent,
    ListsComponent,
    MessagesComponent,
    MemberCardComponent,
    MemberDetailComponent,
    MemberEditComponent,
    PhotoEditorComponent,
    MemberMessagesComponent,
    TaskListComponent,
    TaskDetailComponent,
    TaskCommentsComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    BsDropdownModule.forRoot(),
    BsDatepickerModule.forRoot(),
    PaginationModule.forRoot(),
    TabsModule.forRoot(),
    ButtonsModule.forRoot(),
    ProgressbarModule.forRoot(),
    RouterModule.forRoot(appRoutes),
    NgxGalleryModule,
    FileUploadModule,
    TimeagoModule.forRoot(),
    JwtModule.forRoot({
      config: {
        tokenGetter,
        whitelistedDomains: ['localhost:5000'],
        blacklistedRoutes: ['localhost:5000/api/auth']
      }
    })
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}

```

```

        }
    })
],
providers: [
    AuthService,
    ErrorInterceptorProvider,
    AlertifyService,
    AuthGuard,
    UserService,
    MemberDetailResolver,
    MemberListResolver,
    MemberEditResolver,
    PreventUnsavedChanges,
    ListResolver,
    MessagesResolver,
    TaskListResolver
],
bootstrap: [
    AppComponent
]
}))
export class AppModule { }

```

Приложение Л

Листинг кода юнит-теста для модуля работы с задачами
(рекомендуемое)

```

using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TaskForce.Models;
using TaskForce.Repositories;
using System.Linq;

namespace TaskForce.Tests.TaskService
{
    [TestClass]
    public class TaskServiceTests
    {
        public Mock<TaskRepository> _moqTaskRepo = new
Mock<TaskRepository>();
        public Services.TaskService taskService;
        public List<TaskModel> AllTasks = new List<TaskModel>
        {
            new TaskModel { Guid = Guid.NewGuid(), Name =
"Create" },
            new TaskModel(),
            new TaskModel()
        };

        [TestInitialize]
        public void Initialize()
        {
            AllTasks = new List<TaskModel>
            {
                new TaskModel { Guid = Guid.NewGuid(), Name =
"Create" },
                new TaskModel(),
                new TaskModel()
            };
            _moqTaskRepo.Setup(s =>
s.GetAll()).Returns(AllTasks);
            taskService = new
Services.TaskService(_moqTaskRepo.Object);
        }

        [TestMethod]
        public void Tasks_GetAll()
        {
            var result = taskService.GetAll();
            Assert.AreEqual(3, result.Count);
        }

        [TestMethod]
        public void Tasks_GetAllByName()
        {
            var result = taskService.GetAllByName("Create");
            Assert.AreEqual(1, result.Count);
        }

        [TestMethod]

```

```

        public void Tasks_GetTask()
        {
            var result =
taskService.GetTask(AllTasks.FirstOrDefault().Guid);
            Assert.AreEqual("Create", result.Name);
        }

[TestMethod]
public void Tasks_UpdateTask_Correct()
{
    var newModel = new TaskModel
    {
        Name = "Updated"
    };
    var result =
taskService.UpdateTask(AllTasks.FirstOrDefault().Guid,
newModel);
    var task = taskService.GetAll().FirstOrDefault();
    Assert.IsTrue(result.Success);
    Assert.AreEqual("Updated", task.Name);
}

[TestMethod]
public void Tasks_UpdateTask_Deleted()
{
    foreach (var x in AllTasks)
    {
        x.Deleted = true;
    }
    var newModel = new TaskModel
    {
        Name = "Deleted"
    };
    var result =
taskService.UpdateTask(AllTasks.FirstOrDefault().Guid,
newModel);
    var task = taskService.GetAll().FirstOrDefault();
    Assert.IsFalse(result.Success);
    Assert.AreNotEqual("Deleted", task.Name);
}

[TestMethod]
public void Tasks_UpdateTask_NonExisting()
{
    var newModel = new TaskModel
    {
        Name = "Fail"
    };
    var result = taskService.UpdateTask(Guid.NewGuid(),
newModel);
    Assert.IsFalse(result.Success);
}

[TestMethod]
public void Tasks_UpdateTask_IncorrectParameters()

```

```

        {
            var newModel = new TaskModel
            {
                Name = "          "
            };
            var result =
taskService.UpdateTask(AllTasks.FirstOrDefault().Guid,
newModel);
            var task = taskService.GetAll().FirstOrDefault();
            Assert.IsFalse(result.Success);
            Assert.AreNotEqual("          ", task.Name);
        }

[TestMethod]
public void Tasks_CreateTask()
{
    var newModel = new TaskModel
    {
        Guid = Guid.NewGuid(),
        Name = "NEW",
        Deleted = false
    };
    var result = taskService.CreateTask(newModel);
    Assert.IsTrue(result.Success);
}

[TestMethod]
public void Tasks_DeleteTask_Correct()
{
    var result =
taskService.DeleteTask(AllTasks.FirstOrDefault().Guid);
    Assert.IsTrue(result.Success);
}

[TestMethod]
public void Tasks_DeleteTask_Deleted()
{
    var result = taskService.DeleteTask(Guid.NewGuid());
    Assert.IsFalse(result.Success);
}
}
}

```

Приложение М

Листинг кода сервиса аутентификации `auth.service.ts`

(рекомендуемое)

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { map } from 'rxjs/operators';
import { BehaviorSubject } from 'rxjs';
import { JwtHelperService } from '@auth0/angular-jwt';
import { environment } from 'src/environments/environment';
import { User } from '../_models/user';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  baseUrl = environment.apiUrl + 'auth/';
  jwtHelper = new JwtHelperService();
  decodedToken: any;
  currentUser: User;
  photoUrl = new
BehaviorSubject<string>('../../assets/user.png');
  currentPhotoUrl = this.photoUrl.asObservable();

  constructor(private http: HttpClient) { }

  changeMemberPhoto(photoUrl: string) {
    this.photoUrl.next(photoUrl);
  }

  login(model: any) {
    return this.http.post(this.baseUrl + 'login', model)
      .pipe(
        map((response: any) => {
          const user = response;
          if (user) {
            localStorage.setItem('token', user.token);
            localStorage.setItem('user',
JSON.stringify(user.user));
            this.decodedToken =
this.jwtHelper.decodeToken(user.token);
            this.currentUser = user.user;
            this.changeMemberPhoto(this.currentUser.photoUrl);
          }
        })
      );
  }

  register(user: User) {
    return this.http.post(this.baseUrl + 'register', user);
  }

  loggedIn() {
    const token = localStorage.getItem('token');
    return !this.jwtHelper.isTokenExpired(token);
  }
}

```

Приложение Н
Распределение трудовых ресурсов
(рекомендуемое)

Таблица Н.1 – Распределение трудовых ресурсов

Название задачи	Затраты времени
Анализ предметной области	64 ч
Исследование вариантов использования	16 ч
– Руководитель проекта	16 ч
Обзор существующих решений	8 ч
– Руководитель проекта	8 ч
Выбор программно-аппаратных средства	40 ч
– Руководитель проекта	8 ч
– Программист (backend)	8 ч
– Программист (frontend)	8 ч
– Тестировщик	8 ч
– Дизайнер	8 ч
Техническое задание	48 ч
Анализ требований к системе	32 ч
– Руководитель проекта	16 ч
– Дизайнер	16 ч
Составление технического задания	16 ч
– Руководитель проекта	16 ч
Проектирование и разработка приложения	128 ч
Создание макета	32 ч
– Дизайнер	16 ч
– Программист (frontend)	16 ч
Кодирование серверной части	56 ч
– Программист (backend)	56 ч
Кодирование клиентской части	40 ч
– Программист (frontend)	40 ч
Кодирование завершено	0 ч
Тестирование приложения	72 ч
Поиск ошибок	24 ч
– Тестировщик	8 ч
– Программист (backend)	8 ч
– Программист (frontend)	8 ч
Выявление причин ошибок	16 ч
– Программист (backend)	8 ч
– Программист (frontend)	8 ч
Устранение ошибок	32 ч

– Программист (backend)	16 ч
– Программист (frontend)	16 ч
Тестирование завершено	0 ч
Документация	64 ч
Составление руководства пользователя	12 ч
– Руководитель проекта	12 ч
Составление документации для разработчика	52 ч
– Руководитель проекта	4 ч
– Программист (backend)	16 ч
– Программист (frontend)	16 ч
– Тестировщик	16 ч
Итого	376 ч
Название задачи	Затраты времени

Приложение П
Диаграмма Ганта
(рекомендуемое)

