Министерство образования Республики Беларусь
Учреждение образования
«ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»
Факультет информационных технологий и робототехники
Кафедра «Информационные системы и автоматизация производства»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой

_____ В.Е. Казаков
«___» _____ 2022 г.

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ДИПЛОМНОГО ПРОЕКТА

«Разработка web-сервиса для управления приёмом врача»

Специальность: 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)»

Студент группы Ит-6	_____	М.Л. Лапко
Руководитель	_____	Е.Б. Дунина к.ф.-м.н., доцент
Консультанты:		
экономическая часть	_____	Е.С. Гончарова
охрана труда и промышленная экология	_____	А.В. Гречаников к.т.н., доцент
ресурсосбережение	_____	Е.Б. Дунина к.ф.-м.н., доцент
Нормоконтроль	_____	А.М. Самусев

Витебск, 2022

Учреждение образования
«ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные системы и автоматизация производства»

УТВЕРЖДАЮ
Зав. кафедрой
_____ В.Е. Казаков
«____» _____ 2022 г.

ЗАДАНИЕ
на дипломный проект

Обучающемуся Лапко Максиму Леонидовичу группы Ит-6

1. Тема дипломного проекта Разработка web-сервиса для управления приёмом врача

утверждена приказом ректора университета от 18.04.2022 г. № 197-л

2. Исходные данные к дипломному проекту
Разработать web-сервис для записи на приём к врачу:
Семейство операционных систем: Windows
Программная платформа: ASP.NET Core
Требования:
Регистрация пользователя как доктор, либо пациент;
Возможность пациенту записаться на приём к доктору;
Возможность для доктора редактировать его посещения, завершать их;
Пользователи получают уведомления при оформлении нового посещения, как напоминание о предстоящем посещении;

3. Перечень подлежащих разработке вопросов или краткое содержание расчетно-пояснительной записки
Реферат
Содержание
Введение
1. Анализ объекта
2. Разработка требований
3. Проектирование
4. Реализация
5. Экономическая часть
6. Охрана труда
7. Промышленная экология
8. Ресурсосбережение
Заключение

Список использованных источников

Приложения

4. Перечень графического материала (с точным указанием обязательных чертежей и графиков)

Презентация Microsoft PowerPoint, программа

5. Консультанты по дипломному проекту с указанием относящихся к ним разделов проекта)

Дунина Е.Б. (введение, разделы 1 – 4, 8 заключение)

Гончарова Е. С. (раздел 5)

Гречаников А.В. (разделы 6, 7)

6. Примерный календарный график выполнения дипломного проекта

Раздел 1,2, 3: 25.04.2022 – 28.04.2022

Раздел 4: 29.04.2022 – 13.05.2022

Раздел 5: 14.05.2022 – 18.05.2022

Разделы 6, 7: 19.05.2022 – 25.05.2022

Разделы 8: 26.05.2022 – 01.06.2022

7. Дата выдачи задания 18.04.2020

8. Срок сдачи законченного дипломного проекта 01.06.2020

Руководитель _____ Дунина Е.Б.

Задание принял к исполнению «____» _____ 2022 г.

Подпись обучающегося _____

РЕФЕРАТ

Дипломный проект XX с., XX рис., XX табл., XX источников, XX прил.

ВЕБ-СЕРВИС, REST-API, REACT, ПРИЛОЖЕНИЕ ASP.NET, N-УРОВНЕВАЯ АРХИТЕКТУРА, БАЗА ДАННЫХ, MS SQL SERVER, ТЕСТИРОВАНИЕ, ЭКОНОМИКА, ОХРАНА ТРУДА, ПРОМЫШЛЕННАЯ ЭКОЛОГИЯ, РЕСУРСОСБЕРЕЖЕНИЕ.

Целью разработки является веб-сервис для записи к врачу и управления этими записями.

В процессе разработки проекты были выполнены следующие этапы: проведён анализ предметной области, разработана система требований к системе, описаны аналоги разрабатываемой системы, проведено проектирование структур хранения данных в приложении, разработана архитектура программной системы, а также её компонентов, спроектирован и разработан интерфейс программы, выбраны инструментальные средства для разработки, описаны варианты использования разрабатываемого веб-сервиса, проведено модульное, функциональное и другие виды тестирования, исследованы вопросы экономической эффективности системы, рассмотрены вопросы охраны труда, промышленной экологии и ресурсосбережения.

Областью возможного применения разработанного сервиса являются здравоохранительные организации, такие как: государственные и частные поликлиники и больницы, нуждающиеся в электронном сервисе для записи на приём к специалисту.

Приведённый в дипломном проекте расчётно-аналитический материал

					УО «ВГТУ» ДП.006 1-40 05 01-01 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Лапко М. Л.			РЕФЕРАТ	Лит.	Лист / Листов
Провер.		Дунина Е.Б.					
Реценз.						УО «ВГТУ» каф. ИСАП гр. Ит-6	
Н. Контр.		Самусев А.М.					
Утверд.		Казаков В.Е.					

объективно отражает состояние исследуемого объекта, все заимствования из литературных и других источников теоретические и методологические положения и концепции сопровождаются ссылками на авторов.

ВВЕДЕНИЕ

В настоящее время каждый человек нуждается в периодическом посещении врачей различных специальностей, но сам процесс записи на приём зачастую занимает достаточно большой отрезок времени из-за того, что необходимо дойти до поликлиники/больницы, отстоять всю очередь, которая в большинстве случаев насчитывает от 5 человек, а затем добраться домой. Причём, в итоге может оказаться что требуемый доктор не принимает в удобное вам время или вообще в отпуске.

Конечно, практически во всех современных здравоохранительных учреждениях присутствует система записи посредством звонка, но, как показывает практика, даже на телефонной линии присутствует своеобразная «очередь», когда все телефонные операторы заняты обработкой заявок других пациентов.

При всех вышеописанных обстоятельствах невероятно удобной является система записи на приём посредством использования специализированного веб-сервиса. Бронирование времени для посещения посредством такого веб-сервиса занимает всего от 5 до 15 минут, в зависимости от специфики проблемы пациента и занятости врачей.

Сегодня пользование электронными услугами является простым для большинства современных людей и всё больше людей используют их каждый день. В этом и заключается актуальность разработки сервиса, описанного выше.

Целью дипломного проекта является разработка программной системы «Веб-сервис для бронирования и управления посещениями врача».

Сформулируем основные задачи дипломного проектирования:

- провести анализ предметной области;

- проанализировать и выбрать инструменты разработки программной системы;

- выполнить проектирование системы;

- разработать веб-сервис для бронирования и управления посещениями врача;

- провести тестирование программной системы;

- провести оценку экономической эффективности разработанного приложения;

- провести рассмотрение вопросов охраны труда, экологической безопасности и ресурсосбережения.

1 АНАЛИЗ ОБЪЕКТА

1.1 Описание предметной области

Посещение врача всегда являлось периодической необходимостью каждого отдельного человека, но процесс записи на приём зачастую вызывает определённые затруднения, особенно в последние несколько лет, из-за бушующей пандемии. Также дистанционное бронирование времени на приём рекомендуется из-за возможности передачи заболеваний от человека к человеку при личном общении с работником регистратуры. Телефонные линии поликлиник зачастую перегружены входящими звонками и поэтому довольно трудно дозвониться и выбрать время для посещения специалиста. В связи со всем вышеуказанным мной и была выбрана разработка информационного сервиса по организации именно этой области.

Для анализа избранной области рассмотрим диаграмму прецедентов (рисунок 1.1).

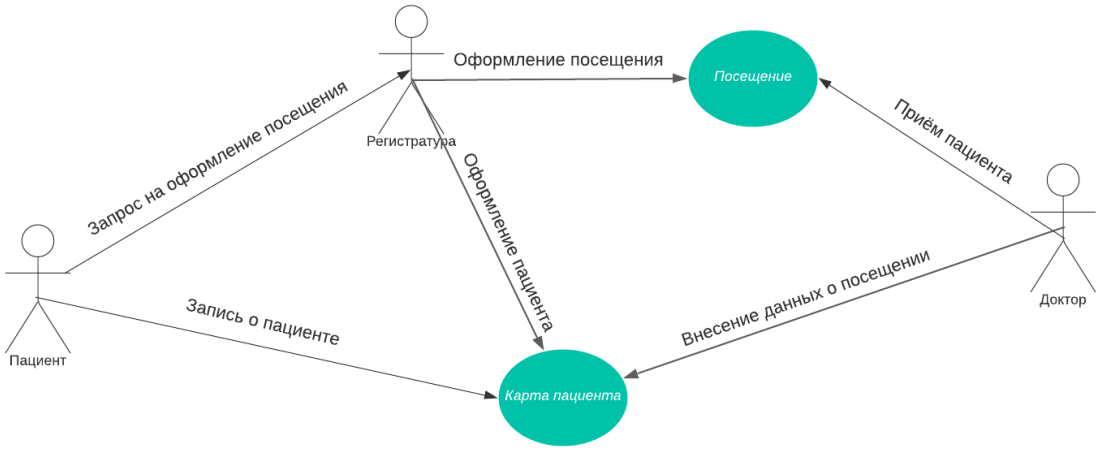


Рисунок 1.1 - Диаграмма прецедентов

					УО «ВГТУ» ДП.006 1-40 05 01-01 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Лапко М. Л.			АНАЛИЗ ОБЪЕКТА	Лит. / Лист / Листов
Провер.		Дунина Е.Б.				
Реценз.						
Н. Контр.		Самусев А.М.				УО «ВГТУ» каф. ИСАП гр. Ит-6
Утверд.		Казаков В.Е.				

Как видно на диаграмме, в рассматриваемой области присутствует 3 сущности: пациент, регистратура, доктор. На первый взгляд взаимодействия между сущностями довольно просты и интуитивно понятны. Единственное, что делает пациент - это подача заявления на посещение врача. У регистратуры же, наоборот, наибольшая роль: оформление и подтверждение посещения, а также оформление карты пациента, которая содержит всю необходимую информацию о пациенте, а также все посещения им специалистов. Доктор, непосредственно, принимает пациентов, то есть, обрабатывает посещения, в итоге он вносит данные о посещении в карту пациента.

Карты пациентов при всём этом обычно представлены лишь в физическом воплощении и не оцифровываются в электронный вариант, таким образом при утере карты невозможно восстановить данные. Учитывая это снова становится очевидным преимущество электронного варианта записи на приём, где карта пациента представлена строкой в базе данных.

1.2 Построение концептуальной модели предметной области

Диаграмма последовательностей разрабатываемой системы представлена ниже (рисунок 1.2).

Можно увидеть, что изначально после входа на сайт пользователю необходимо авторизоваться, если пользователь не имеет аккаунта, то он может зарегистрироваться, причём при регистрации пользователь может зарегистрироваться в качестве доктора или пациента. Сама регистрация доктора отличается от регистрации пациента наличием полей для информации, специфической для специалиста, например, номер врачебной лицензии или специализация.

После авторизации/регистрации пользователь перенаправляется на главную страницу, которая отличается для доктора и пациента. Доктор может посетить свой профиль для редактирования информации, просмотреть

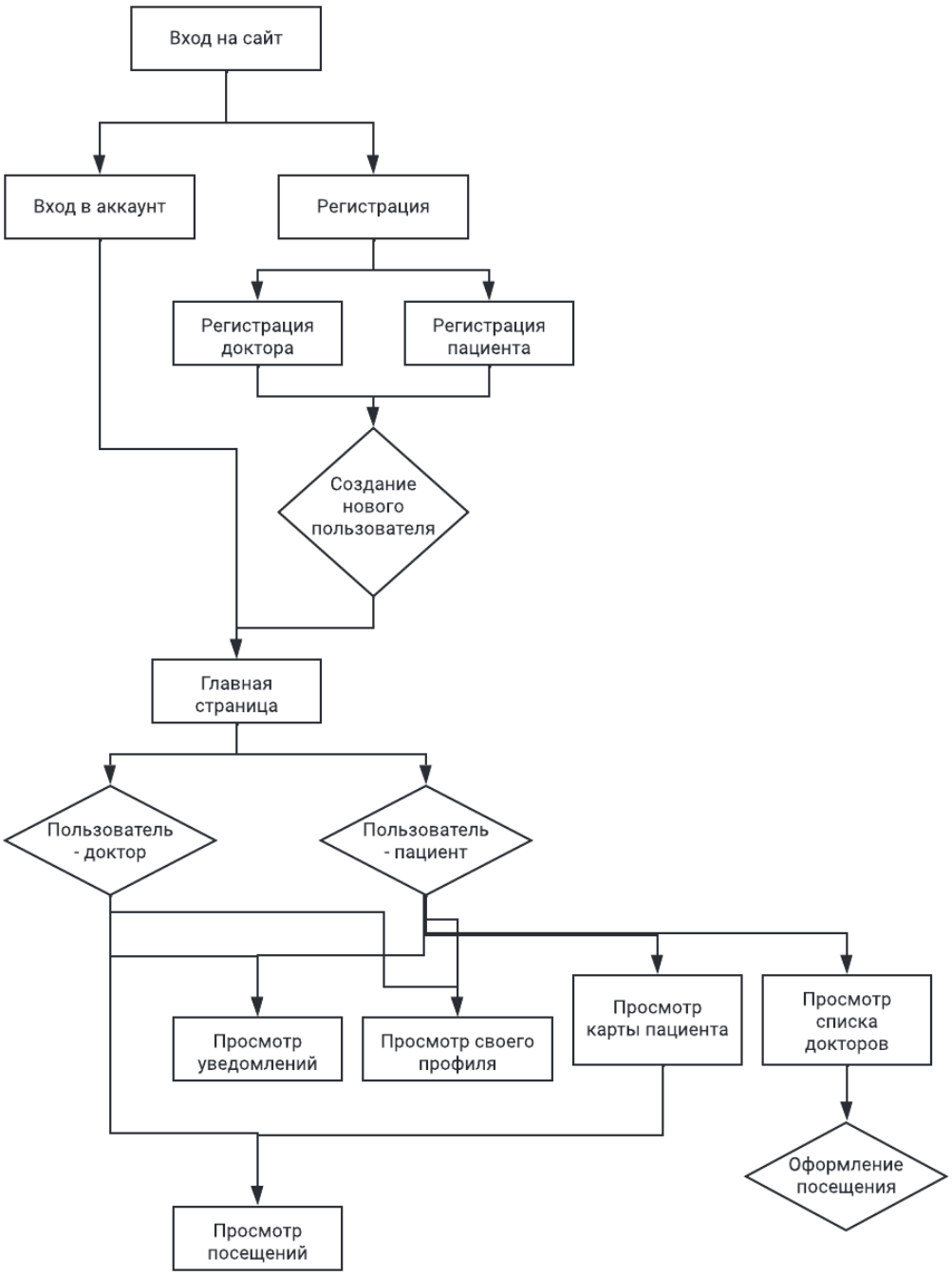


Рисунок 1.2 - Диаграмма последовательностей

уведомления, которые приходят при оформлении нового посещения к данному специалисту. Также доктор может просмотреть предстоящие для него посещения, каждое посещение содержит в себе ссылку на карту пациента, которую доктор также может просмотреть для ознакомления с

пациентом и его заболеваниями. Доктор может редактировать информацию о посещении, например, указывать заболевание, которое было найдено у пациента в результате приёма или добавлять определённое описание на своё усмотрение.

Пациент же имеет доступ к списку всех докторов, которых он может фильтровать различными способами: по специализации, опыту, фамилии, городу работы и так далее. Выбрав предпочтительного специалиста, пациент может оформить заявку на посещение.

Подобно доктору пациент может просмотреть свой профиль и свободно его редактировать, также пациент может напрямую просмотреть свою карту, которая содержит записи обо всех его посещениях. Также пациент может просмотреть свои уведомления, которые приходят при подтверждении его заявки на посещение и как напоминание о предстоящем посещении.

Можно заметить, что, в отличие от диаграммы прецедентов (рисунок 1.1) в концептуальной модели отсутствует регистратура. В отличие от реальной системы записи на приём, в веб-сервисе роль регистратуры частично исполняет сама система, сохраняя все данные и уведомляя пользователей, частично сам пациент, оформляя заявку на посещение и частично доктор, утверждая заявку.

2 РАЗРАБОТКА ТРЕБОВАНИЙ

2.1 Определение требований к программной системе

Требования определяются в зависимости от роли пользователя, так как пользователи разделяются на докторов и пациентов. Таким образом определим требования, разделяющие возможностей работы различных типов пользователей с информационной системой.

Возможности доктора:

- просмотр своего профиля;
- возможность редактирования профиля;
- возможность удаления профиля;
- просмотр уведомлений;
- просмотр предстоящих посещений;
- возможность отклонить входящее посещение;
- возможность редактирования самого посещения.

Возможности пациента:

- просмотр своего профиля;
- возможность редактирования профиля;
- возможность удаления профиля;
- просмотр уведомлений;
- просмотр списка докторов;
- возможность оформить запрос на посещение специалиста;
- просмотр своей карты;
- просмотр своих посещений.

Дополнительные требования:

- возможность новому пользователю зарегистрироваться как доктор или пациент;

- доктору приходит уведомление при записи нового пациента к нему на приём, пациенту приходит уведомление, когда доктор подтверждает посещение или как напоминание о предстоящем посещении;

- подтверждение электронной почты для каждого пользователя путём письма на указанную пользователем почту;

- возможность смены пароля путём перехода по ссылке из письма, присылаемого на подтверждённую электронную почту пользователя.

2.2 Описание аналогов системы

В интернете существует множество примеров систем онлайн записи к доктору, практически у каждой поликлиники присутствует официальный сайт, где можно записаться на приём, также существуют сервисы, не относящиеся к какому-либо определённому медицинскому учреждению. Примеры таких систем:

- talon.by - самый популярный сервис по оформлению посещения онлайн в Беларуси;

- 2doc.by - сервис для записи на приём в большом количестве регионов Беларуси;

- iBolit.pro - сервис телемедицины;

- yclients.com - онлайн-запись и автоматизация процессов клиники.

Представленные сервисы имеют возможность выбора определённого учреждения здравоохранения, чтобы впоследствии предоставить выбор среди специалистов, работающих в этом учреждении. Некоторые сервисы дополнительно позволяют заранее выбрать предпочтительную область страны чтобы затем выбрать поликлинику.

Также данные интернет-ресурсы в большинстве своём имеют дополнительные возможности, зачастую относящиеся к определённой

поликлинике, например: платные услуги, вызов врача на дом и так далее.

Представленные сервисы также имеют недостатки, как, например, короткий период бесплатного пользования, отсутствие тестового периода и бесплатной версии, узкая направленность приложения, высокая стоимость обслуживания, необходимость платить абонентскую плату для доступа к сервису.

3 ПРОЕКТИРОВАНИЕ

3.1 Проектирование структур хранения данных

Для работы с данными была выбрана Microsoft SQL Server. Microsoft SQL Server - система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов - Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка. SQL Server - это основа платформы обработки данных Майкрософт, которая предоставляет надёжную и устойчивую производительность (в том числе благодаря технологиям обработки данных в памяти) и помогает быстрее извлечь ценную информацию из любых данных, расположенных как в локальной среде, так и в облаке [1].

Схема базы данных, отвечающей за хранение информации в проекте, представлена ниже (рисунок 3.1).

Для комфортной работы с данными в процессе разработки и тестирования используется среда SQL Server Management Studio.

Среда SQL Server Management Studio - это единая универсальная среда для доступа, настройки и администрирования всех компонентов MS SQL Server, а также для разработки компонентов системы, редактирования текстов запросов, создания скриптов и пр. Благодаря наличию большого количества визуальных средств управления, среда SQL Server Management Studio

позволяет выполнять множество типовых операций по администрированию MS SQL Server администраторам с любым уровнем знаний SQL Server. Удобная среда разработки, встроенный веб-браузер для быстрого обращения к библиотеке MSDN или получения справки в сети, подробный учебник, облегчающий освоение многих новых возможностей, встроенная справка от сообществ в Интернете и многое другое позволяют максимально облегчить

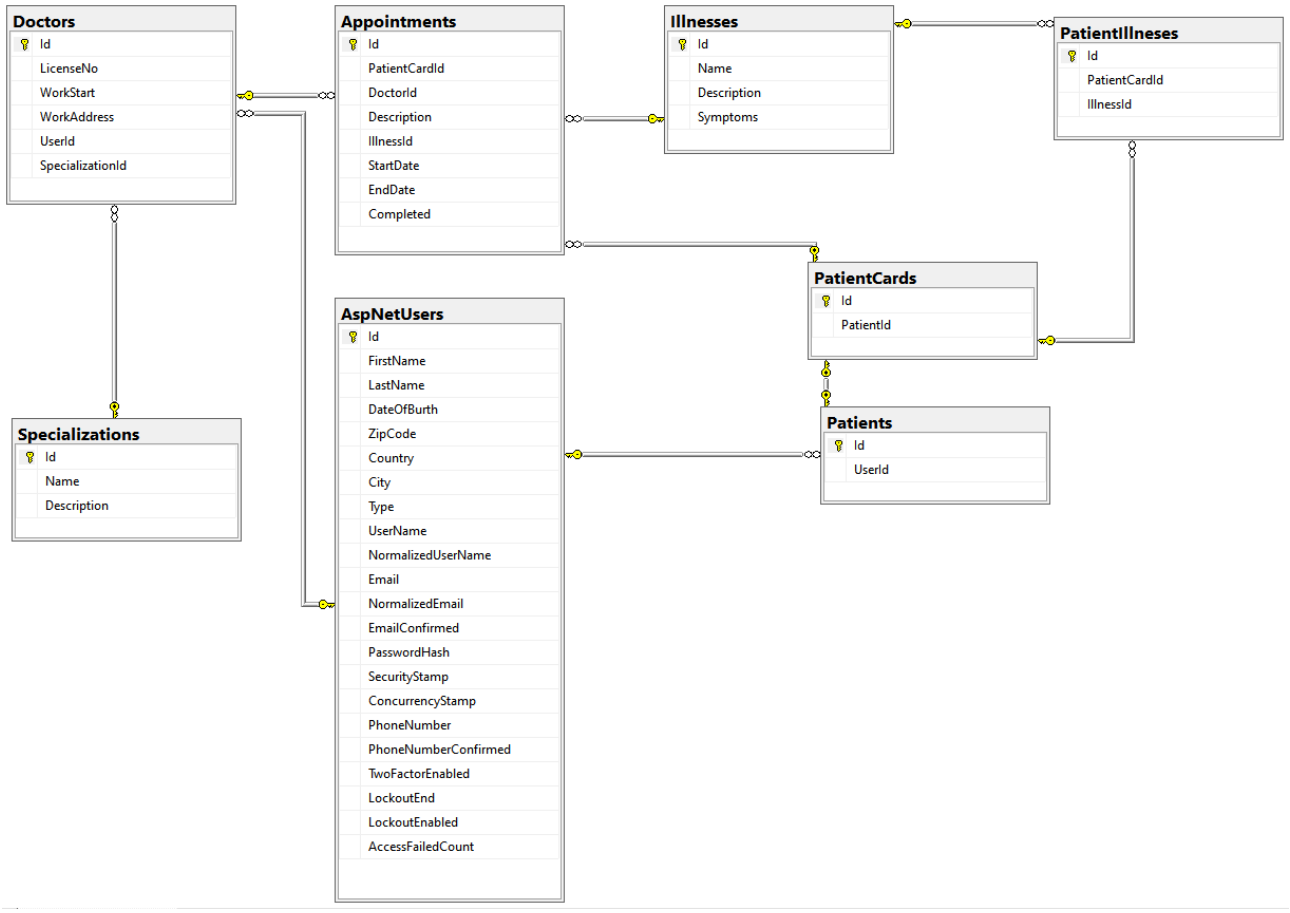


Рисунок 3.1 - Схема базы данных

процесс разработки в среде SQL Server, а также даёт богатые возможности для создания различных сценариев SQL Server [2].

Также, в рамках данного раздела, стоит упомянуть о способе хранения данных в фронтэнд части приложения. За это отвечает Redux.

Redux - это инструмент для управления состоянием данных и пользовательским интерфейсом в приложениях JavaScript с большим

количеством сущностей. Представляет собой библиотеку JavaScript. Название читается как «Редакс» и составлено из двух слов: reduce и flux. Reduce - это функция, которая приводит большую структуру данных к одному значению. Flux - архитектура приложения, при которой данные передаются в одну сторону. Инструмент основан на этих двух понятиях, поэтому они вынесены в название. Обычно Redux используется в связке с фреймворками для JavaScript: React, TypeScript, Vue, Angular и другими. Реже он бывает нужен для написания кода на чистом JS. Имеет открытый исходный код и доступен бесплатно. Со всеми зависимостями весит всего около 2 Кб.

Для чего нужен Redux:

- для управления состоянием приложения, работающего с большим количеством данных;
- для удобной замены встроенных средств работы с состоянием в React;
- для более лёгкого масштабирования приложения, его преобразования под разные задачи;
- для избавления от ошибок, связанных с беспорядком в объекте состояния;
- для предсказуемости и понятности работы приложения;
- для более простой отладки и доработки;
- для повышения производительности и работоспособности программы [3].

3.2 Разработка архитектуры программной системы

Приложение состоит из 2 частей: бекэнд и фронтэнд. Они общаются между собой посредством протоколов HTTP и HTTPS. Пользователь взаимодействует лишь с фронтэнд частью, которая предоставляет ему интерфес приложения, взаимодействия с которым запускают запросы к бекэнд части приложения.

Схема развёртывания приложения представлена ниже (рисунок 3.2).

Для разработки бэкэнд части проекта была выбрана многоуровневая архитектура (N-layer).

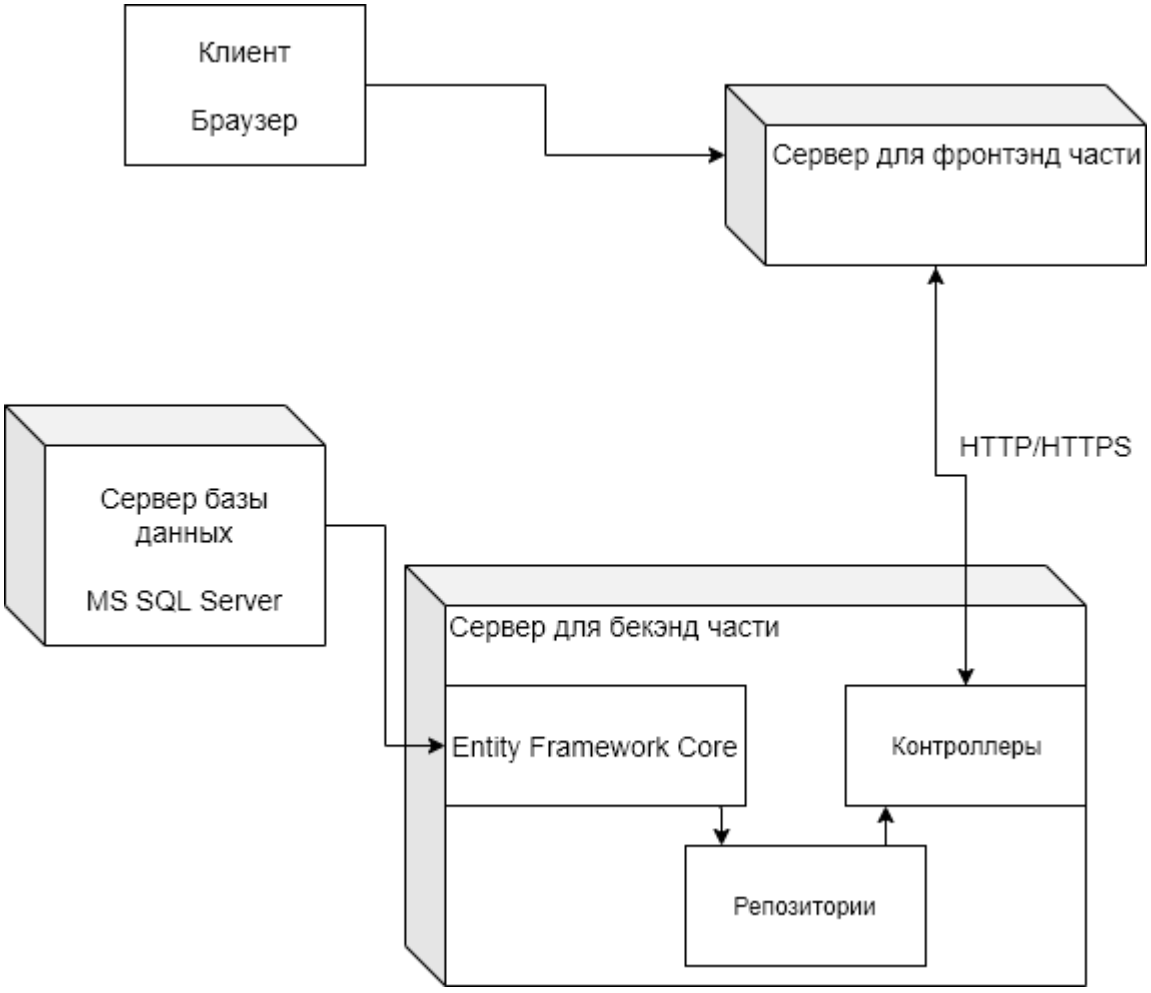


Рисунок 3.2 - Диаграмма развёртывания приложения

N-уровневая архитектура - это концепция клиент-серверной архитектуры в программной инженерии, в которой функции представления, обработки и управления данными логически и физически разделены. Каждая из этих функций работает на отдельном компьютере или в отдельных кластерах, так что каждая из них может предоставлять услуги с максимальной пропускной способностью, поскольку отсутствует совместное использование ресурсов.

Такое разделение делает управление каждым отдельным ресурсом проще, так как выполнение работы над одним не влияет на другие, изолируя любые проблемы, которые могут возникнуть.

N-уровневая архитектура обычно делит приложение на три уровня: уровень представления, логический уровень и уровень данных. Это физическое разделение различных частей приложения в отличие от обычно концептуального или логического разделения элементов в структуре модель-представление-контроллер (MVC). Другое отличие от инфраструктуры MVC состоит в том, что n-уровневые уровни связаны линейно, то есть вся связь должна проходить через средний уровень, который является логическим уровнем. В MVC нет реального среднего слоя, потому что взаимодействие является треугольным; уровень управления имеет доступ как к слоям вида, так и к слою модели, а модель также обращается к виду; Контроллер также создаёт модель на основе требований и передаёт её в представление. Однако они не являются взаимоисключающими, поскольку инфраструктура MVC может использоваться в сочетании с n-уровневой архитектурой, причём n-уровень является общей используемой архитектурой, а MVC используется в качестве основы для уровня представления [4].

Данная архитектура была выбрана так как имеет следующие преимущества:

- более простая реализация по сравнению с другими подходами;
- предлагает абстракцию благодаря разделению ответственностей между уровнями;
- изолирование защищает одни слои от изменений других;
- повышает управляемость программного обеспечения за счёт слабой связанности [5].

Схема многоуровневой архитектуры представлена ниже (рисунок 3.3).

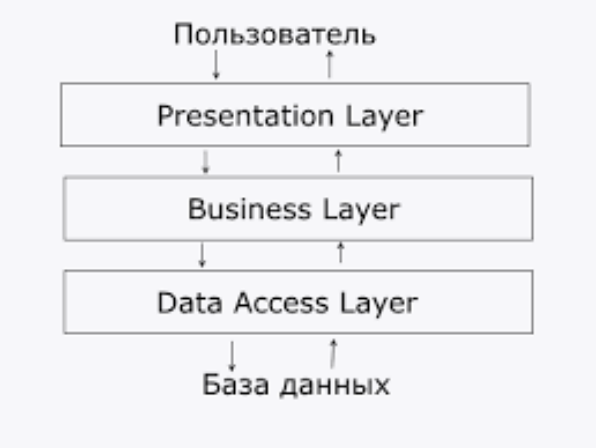


Рисунок 3.3 - Схема связей между слоями многоуровневой архитектуры

3.3 Разработка архитектуры компонентов программной системы

Приложение состоит из двух компонентов: бекэнд и фронтэнд.

В бэкэнд части приложения представлены 4 «слоя»:

- API - слой. Так как бэкэнд часть приложения представляет собой REST API, этот слой играет роль представления в классической N-layer архитектуре;

- слой бизнес-логики. Это библиотека классов, содержащая классы, необходимые для различных вычислений и преобразований, например, AutoMapper, который служит для автоматической трансляции объекта одного типа в объект другого типа;

- слой моделей. Это библиотека классов, содержащая классы, представляющие собой объекты и сущности системы;

- слой доступа к данным. Это библиотека классов, содержащая всё необходимое для работы с базой данных: контекст - позволяет работать с БД, репозитории - классы, работающие с определёнными частями контекста для упрощённого доступа к данным, миграции - записи, диктующие как правильно транслировать код в базу данных и наоборот.

Как видно, в отличие от типичной N-уровневой архитектуры, которая содержит 3 слоя, приложение имеет 4 слоя. Это потому, что данная

архитектура не устанавливает жёстких правил и позволяет вводить дополнительные слои. Можно сказать, что слой API и слой моделей вместе представляют собой слой представления, диктующий какие данные и как будут отображаться для конечного пользователя.

В обозревателе решений Visual Studio архитектура проекта выглядит следующим образом (рисунок 3.4).

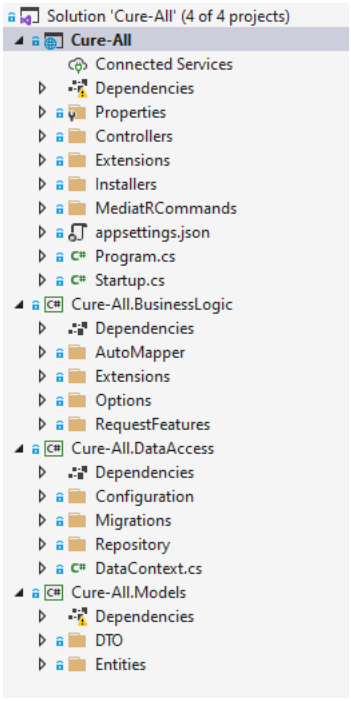


Рисунок 3.4 - Многоуровневая архитектура проекта в Visual Studio

Фронтэнд часть проекта представлена единым React-приложением, которое организовано в стиле многоуровневой архитектуры, слои представлены не отдельными библиотеками классов, а папками с файлами в основном проекте:

- Api - слой доступа к данным - папка содержит TypeSctipt файлы, осуществляющие запросы к бэкэнд части проекта;

- Components - слой представления - папка содержит компоненты представления интерфейса;

- Content - папка содержит файлы, необходимые для построения интерфейса, не являющиеся стилями, например изображения;

- Store - папка содержит все компоненты, необходимые для работы хранилища Redux;

- Styles - папка содержит файлы, содержащие стили для построения пользовательского интерфейса.

В обозревателе Visual Studio Code структура приложения выглядит следующим образом (рисунок 3.5).

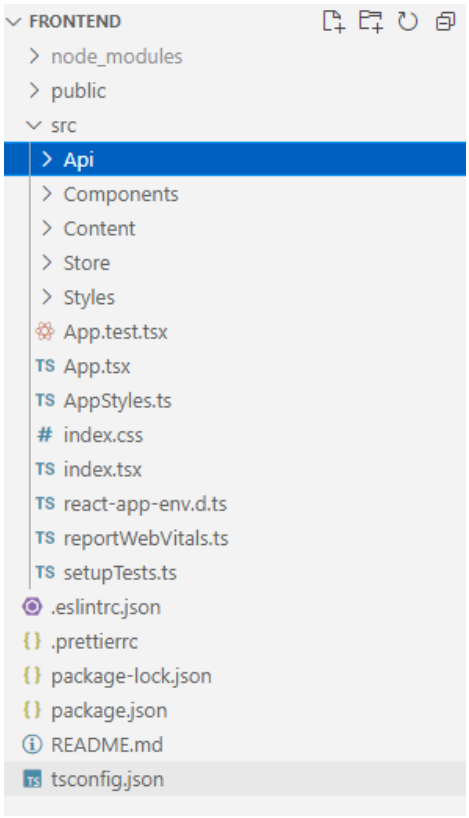


Рисунок 3.5 - Структура приложения в Visual Studio Code

Фронтэнд часть отвечает лишь за получение и отображение данных, поэтому не имеет привычной для многоуровневой архитектуры слой бизнес-логики.

3.4 Разработка интерфейса программного продукта

Фронтэнд часть приложения предоставляет графический пользовательский интерфейс(GUI). Как упоминалось в предыдущих разделах, фронтэнд часть создана с использованием библиотеки для JavaScript React с поддержкой TypeScript. Компоненты GUI представляют собой классовые компоненты React, маршрутизация между страницами реализована с использованием библиотеки React-Router. Для стилизации интерфейса использован стандартный JavaScript фреймворк Bootstrap.

Bootstrap - это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрого создания адаптивных дизайнов сайтов. Фреймворк Bootstrap используется не только независимыми разработчиками, но и целыми компаниями. Основная область его применения – это разработка фронтенд составляющих сайтов и интерфейсов админок. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным. В сущности, Bootstrap - это просто набор файлов (CSS и JavaScript). После подключения этих файлов к странице вам станут доступны для вёрстки дизайна большое количество классов и готовых компонентов. Используя их можно очень быстро и качественно создать современный адаптивный дизайн сайта [6].

Bootstrap позволяет легко позиционировать элементы интерфейса, так как имеет, так называемую систему «сетки». Вся страница представляет собой одну большую матрицу, которая по умолчанию состоит из строк по 12 колонок. Регулируя размер компонентов интерфейса можно помещать до 12 элементов в одной строке, причём каждый элемент также представляет из себя строку из 12 колонок. Таким образом, используя вложенные друг в друга компоненты можно легко создать необходимый интерфейс.

За CSS стилизацию отвечают библиотеки Emotion и Styled.

Библиотека Emotion позволяет создавать CSS-классы так, что возможные повторяющиеся имена классов на разных уровнях интерфейса не будут препятствием. Имя CSS-класса формируется из названия самого класса, названия компонента, в котором он используется, а также особого, уникального, идентификатора. Таким образом можно выносить CSS-классы в отдельные файлы и не заботиться об уникальности их имён.

Библиотека Styled даёт возможность создавать собственные компоненты, опираясь на основные компоненты, например кнопка, button, присваивая новому компоненту имя, а также добавляя к нему CSS-стилизацию, можно затем использовать его как обычный компонент, например button.

Для валидации форм интерфейса, например, если неверно написан пароль, используется библиотека React-Hook-Form. Данная библиотека позволяет легко регистрировать определённые компоненты формы для определённых полей заполняемых форм, а также задавать правила валидации напрямую в этих компонентах, например, чтобы значение не превышало 10. Также позволяет установить определённые сообщения об ошибках для каждого правила валидации каждого поля и выводить эти сообщения если они появляются. С использованием TypeScript, использование React-Hook-Form становится ещё проще, так как можно заранее определить интерфейс для объекта, который будет заполнятся в форме и объявить React-Hook-Form что объект именно этого интерфейса заполняется в форме, что, например, заранее позволяет узнать тип данных определённого поля.

Интерфейс адаптивен под различные размеры экрана, так как Bootstrap предоставляет возможность указать размеры компонентов для различных размеров экрана устройства клиента.

Фронтэнд часть приложения содержит большое количество страниц, схема переходов между ними отличается для пациента и доктора из-за того, что они имеют доступ к разному функционалу приложения.

Схема переходов между страницами для пациента представлена ниже (рисунок 3.6).

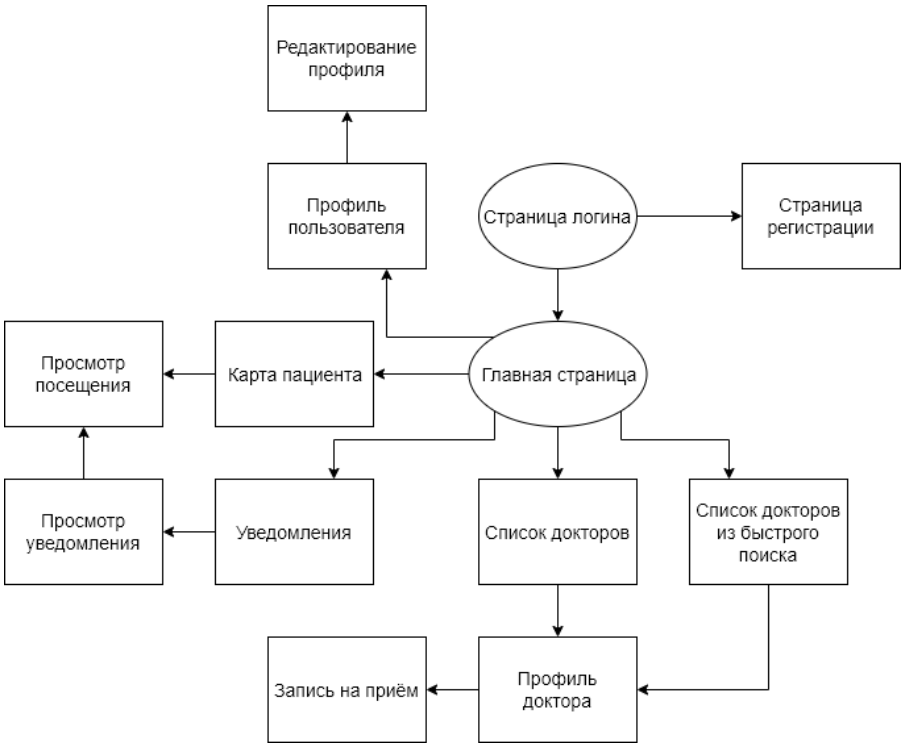


Рисунок 3.6 - Схема переходов между страницами для пациента

Благодаря навигационной панели пациент имеет доступ, например, к списку докторов, не только с главной страницы. Навигационная панель и заголовок сайта находятся на любой странице, таким образом, у пользователя всегда имеется доступ на любую из страниц, на которые есть переходы с главной страницы на схеме (рисунок 3.6).

Схема переходов между страницами для доктора представлена ниже (рисунок 3.7).

Доктор имеет немного отличный от пациента функционал, таким образом схема переходов между страницами отличается. Также, как и пациент, доктор имеет возможность переадресации при помощи навигационной панели и заголовка сайта.

Данные в приложении в большинстве случаев представлены в виде

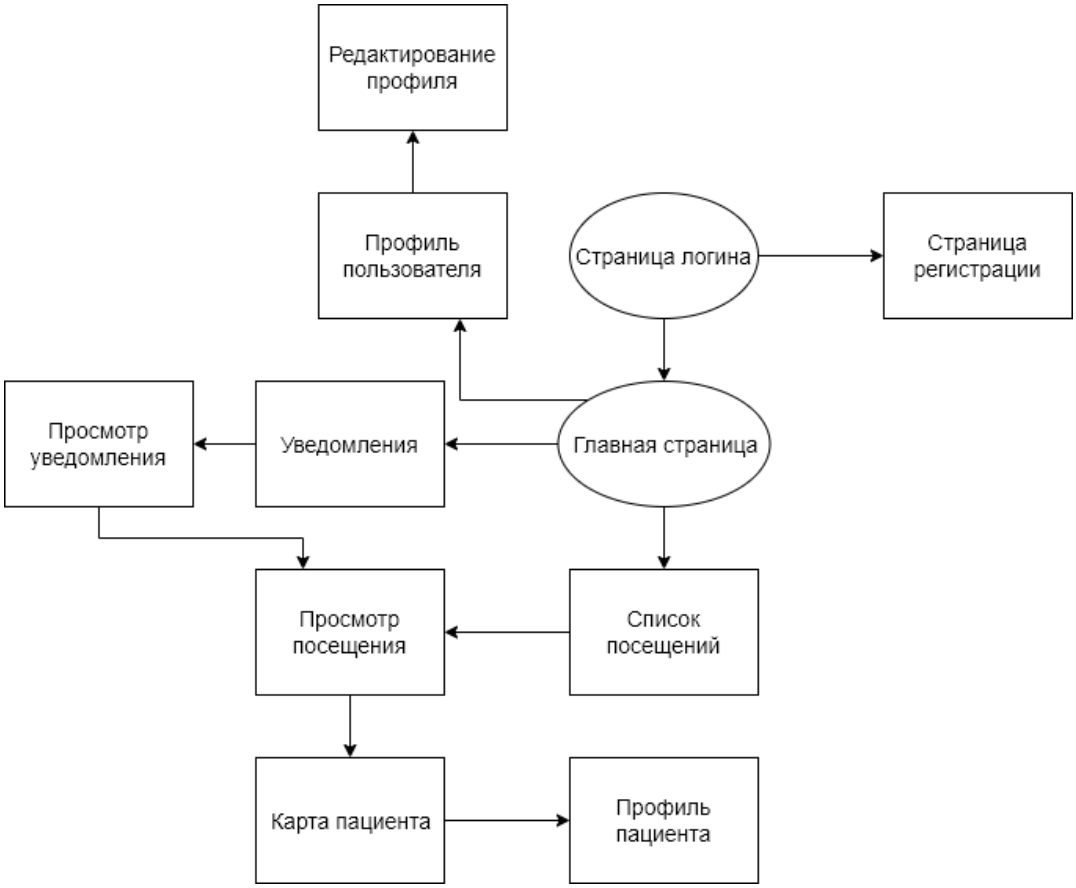


Рисунок 3.7 - Схема переходов между страницами для доктора

списка, например, список докторов, пример структуры списка в приложении представлен ниже (рисунок 3.8).

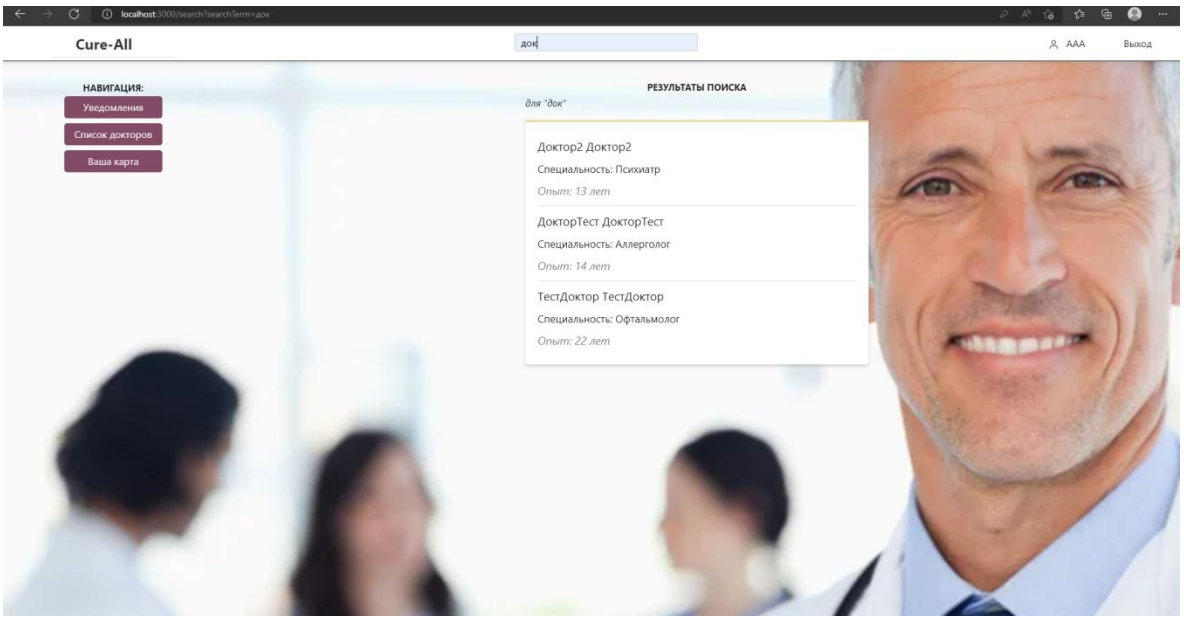


Рисунок 3.8 - Пример списка докторов в приложении

Также, любые данные из списка можно просмотреть индивидуально, например, профиль отдельного доктора из списка, пример представления отдельной сущности из списка представлен на рисунке 3.9.

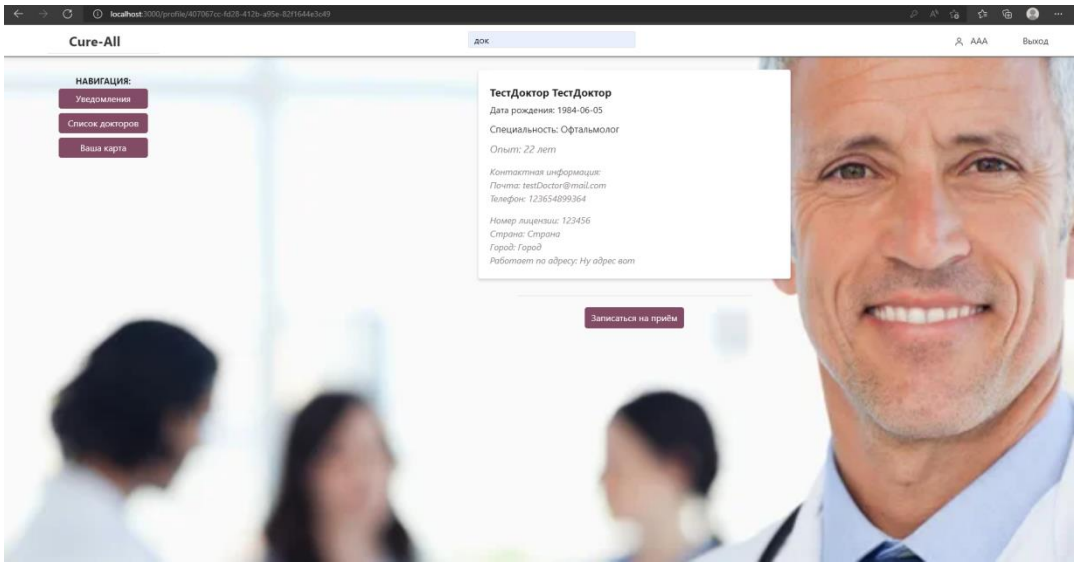


Рисунок 3.9 - Пример представления профиля доктора

Интерфейс приложения имеет функцию вывода окна подтверждения определённых действий, например, удаления аккаунта пользователя. Пример окна подтверждения представлен ниже (рисунок 3.10).

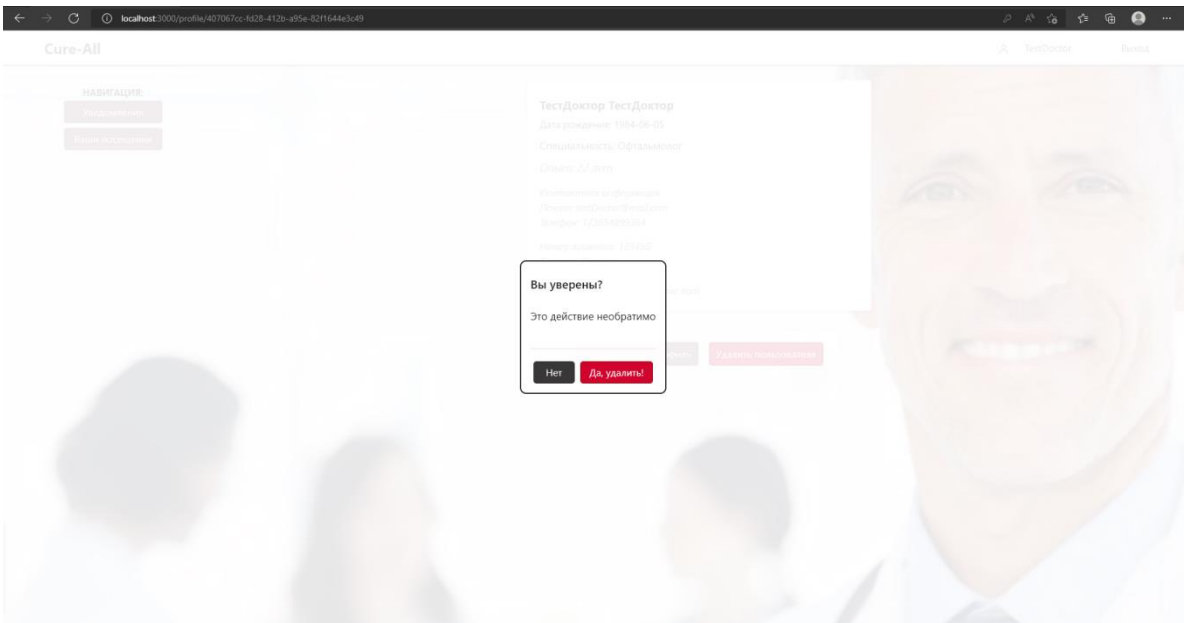


Рисунок 3.10 - Пример окна подтверждения

4 РЕАЛИЗАЦИЯ

4.1 Выбор инструментальных средств, системного и дополнительного программного обеспечения

Приложение, разрабатывающееся в рамках дипломного проектирования, состоит из двух частей: бэкэнд и фронтэнд.

Для разработки бэкэнд части информационной системы был выбран язык C#. C# - это современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надёжных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# - объектно-ориентированный, ориентированный на компоненты язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# - объектно-ориентированный язык. Вы определяете типы и их поведение.

Вот лишь несколько функций языка C#, которые позволяют создавать надёжные и устойчивые приложения. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения

поддерживают приёмы функционального программирования. Синтаксис LINQ создаёт общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В C# имеется Единая система типов. Все типы C#, включая типы-примитивы, такие, как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощённые структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода [7].

Язык C# практически универсален. Можно использовать его для создания любого ПО: продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android.

C# популярен за счёт своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

Популярность языка - ещё одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке Microsoft. Программисты, хорошо знакомые с C#, востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество [8].

Также в языке присутствует обилие синтаксического сахара, который делает тяжёлую жизнь программиста капельку слаще. Вместо того чтобы писать 100500 строк кода, присутствует возможность использовать готовую конструкцию, а компилятор сделает всю остальную работу. Но некоторые такие конструкции являются не самыми оптимальными с точки зрения производительности. Но все это перекрывается за счёт удобочитаемости кода и высокой скоростью разработки [9].

Сама разработка бэкэнд части выполнена при помощи платформы ASP.NET Core.

ASP.NET Core - это веб-инфраструктура с открытым исходным кодом, оптимизированная для облачных вычислений, для разработки современных веб-приложений, которые можно разрабатывать и запускать на Windows, Linux и Mac. Он включает в себя инфраструктуру MVC, которая теперь объединяет функции MVC и веб-API в единую среду веб-программирования. Он был переработан с нуля, чтобы быть быстрым, гибким, современным и работать на разных платформах. В дальнейшем ASP.NET Core - это фреймворк, который можно использовать для веб-разработки с .NET. Также ASP.NET Core имеет схожие функции с MVC и веб-API [10].

Бэкэнд часть является веб-API. Платформа веб-API ASP.NET позволяет с легкостью создавать службы HTTP для широкого диапазона клиентов, включая браузеры и мобильные устройства. ASP.NET Web API - это идеальная платформа для сборки REST-приложений на базе .NET Framework [11].

Для написания приложения была выбрана среда программирования Microsoft Visual Studio.

Microsoft Visual Studio является средой программирования, разработанной компанией Microsoft. Эта среда позволяет создавать кроссплатформенные проекты на различных языках программирования, таких как Visual Basic, Visual C#, Visual C++, Visual F# и другие. Также она позволяет создавать

программы, использующие в своей работе платформу .NET, которая позволяет использовать большой набор сервисов, реализующихся в виде промежуточного, не зависящего от базовой архитектуры, кода. Основной целью создания платформы .NET является возможность реализации разработчиками специальных сервисно-ориентированных программ, работающих на любых платформах.

MS Visual Studio позволяет разработчику иметь доступ к огромной коллекции различных функций, которые позволяют вести разработки для любой версии операционной системы семейства Windows, для интернет-приложений и мобильных приложений. Также среда программирования открывает широкие возможности в области облачных технологий. Эта среда открывает разработчику широкие возможности для реализации самых разных проектов, реализуя высокую производительность и независимость от особенностей оборудования.

Microsoft Visual Studio позволяет осуществлять проектирование программ, используя любые по размеру команды. Эта среда разработки предоставляет инструменты планирования для возможности внедрения методов последовательной разработки, а также для гибкого планирования. Используя весь спектр возможностей, предоставляемых MS Visual Studio, можно реализовать максимально полную систему, наиболее удачно спроектировать любую архитектуру. Таким образом Microsoft Visual Studio представляет собой передовую среду разработки [12].

Фронтэед часть разработана при помощи JavaScript-библиотеки React при поддержке TypeScript.

React - библиотека, написанная на JavaScript, которая используется для работы с интерфейсами. В 2011 году её начали использовать для социальной сети Facebook, а уже в 2013 году библиотеку выложили в открытый доступ, и энтузиасты со всего мира начали создавать инструменты для расширения её возможностей. Разработчики используют React, чтобы создавать интерфейсы,

которые способны менять контент без перезагрузки страницы. Благодаря этому сайты или нативные приложения быстро отзываются на действия пользователей. Можно добавлять товары в корзину без перезагрузки страницы или заполнять формы без переадресации.

React считается самой популярной библиотекой в мире, написанной на JS. Её популярность обеспечивается тем, что с помощью компонента можно решать разные задачи. Особенно, если использовать дополнительные инструменты, которые интегрируются в проекты и открывают доступ к нестандартным возможностям [13].

React предпочтительнее использовать по следующим причинам:

- технология SPA (single-page application, по-русски: “разработка одностраничных приложений”): React поможет вам создать одностраничное приложение. С помощью ReactJS вы сможете изменять (управлять/манипулировать) контент всей страницы с минимальным кодом;

- декларативный подход: React использует декларативный дизайн со всем синтаксическим сахаром, что помогает написать поддерживаемый код высокого уровня. Вам просто нужно определить цель, и React будет обрабатывать инструкции JavaScript DOM с учётом ситуаций, в которых они используются;

- компонент-управляемый пользовательский интерфейс: React основан на компонентной концепции. Компоненты являются многоразовыми строительными блоками в пользовательском интерфейсе. С помощью ReactJS вы можете создать инкапсулированный компонент, который управляет своими данными, и избежать сценария влияния на другие состояния и действия компонентов в DOM-дереве. Это только одна из характеристик компонент-управляемого интерфейса. Он также помогает повторно использовать код, разделять ответственность и избегать повторения [14].

TypeScript (TS, TScript или «тайпскрипт») - это язык программирования для веб-разработки, основанный на JavaScript. Делает код понятнее и

надёжнее, добавляет статическую типизацию (переменные привязаны к конкретным типам данных), а также может быть скомпилирован в JavaScript. TypeScript используют фронтенд- и бэкенд-разработчики.

TypeScript добавляет в язык строгую типизацию. Каждой переменной при создании присваивается определённый тип (type) — стандартный или созданный самим разработчиком. Создать тип можно в пределах возможностей языка: например, число от 1 до 31 для записи дня в месяц или массив из двух элементов для записи координат.

TypeScript помогает сократить время на выявление и устранение багов, которые иногда сложно найти в динамической среде JavaScript. С помощью TypeScript можно написать более понятный и читаемый код, который максимально описывает предметную область. Таким образом архитектура становится более выраженной.

Код, написанный на TypeScript, не выполнится напрямую в браузере. Поэтому TS - не самостоятельный язык, а именно языковая надстройка над JS.

Для его работы нужен дополнительный этап - транспиляция, когда программное обеспечение преобразует написанный на TypeScript код в «чистый» JavaScript.

JS не требует установки в систему: его по умолчанию поддерживает любой браузер. А вот TypeScript понадобится установить, потому что для транспиляции необходим модуль tsc [15].

Код для фронтэнд части приложения написан при помощи редактора кода Visual Studio Code.

Visual Studio Code - это «бесплатный редактор, который помогает программисту писать код, помогает в отладке и исправлении кода с помощью метода IntelliSense». В обычных условиях это облегчает пользователю написание кода простым способом. Многие говорят, что это половина IDE и редактора, но решение остаётся за программистами. Visual Studio Code

поддерживает несколько языков программирования, имеет кроссплатформенную поддержку, огромное количество расширений для упрощения создания кода, а также встроенную поддержку системы контроля версий Git.

Visual Studio Code поддерживает несколько языков программирования. Так что раньше программистам требовалась веб-поддержка: другой редактор для разных языков, но он имеет встроенную многоязычную поддержку. Это также означает, что он легко обнаруживает, если есть какая-либо ошибка или ссылка на другой язык, он сможет легко обнаружить её [16].

4.2 Описание реализации вариантов использования

Вариант использования - это связный блок функциональности, которую предоставляет классификатор (система, подсистема или класс). Этот блок описывает последовательность сообщений, которыми обменивается система и один или несколько внешних пользователей (актантов), а также действия, осуществляемые при этом системой.

Вариант использования служит для определения некой части поведения классификатора (которым можно также считать подсистему и даже всю систему целиком), без указания на его внутреннюю структуру. Каждый вариант использования описывает некую услугу, которую предоставляет своим пользователям классификатор. Иначе говоря, это некоторый способ использования классификатора, который виден со стороны. Вариант использования описывает всю последовательность сообщений, которую начинает пользователь (и модели - актант), в терминах взаимодействия между пользователем и классификатором, включая ответы классификатора. К взаимодействию относятся только коммуникации между системой и актантами. Внутреннее поведение и реализация скрыты. Все множество вариантов использования какого-либо классификатора или системы разделяет и полностью описывает его поведение. Каждый вариант

использования представляет собой некую разумную долю функциональности, которая доступна пользователям. Обратите внимание, что под термином пользователь следует понимать не только людей, но и компьютеры, а также прочие объекты. Актант представляет собой некую идеализацию намерений пользователя, а не самого этого пользователя. Один реальный пользователь может соответствовать нескольким актантам, а один актант может представлять одно и то же намерение сразу нескольких пользователей.

Вариант использования включает в себя описание основного поведения, осуществляемого в ответ на запрос пользователя, а также все возможные варианты этого поведения, например альтернативные последовательности, исключительное поведение и обработка ошибок. Для большего удобства варианты использования можно группировать в пакеты [17].

Диаграмма вариантов использования приложения пользователями представлена ниже (рисунок 4.1).

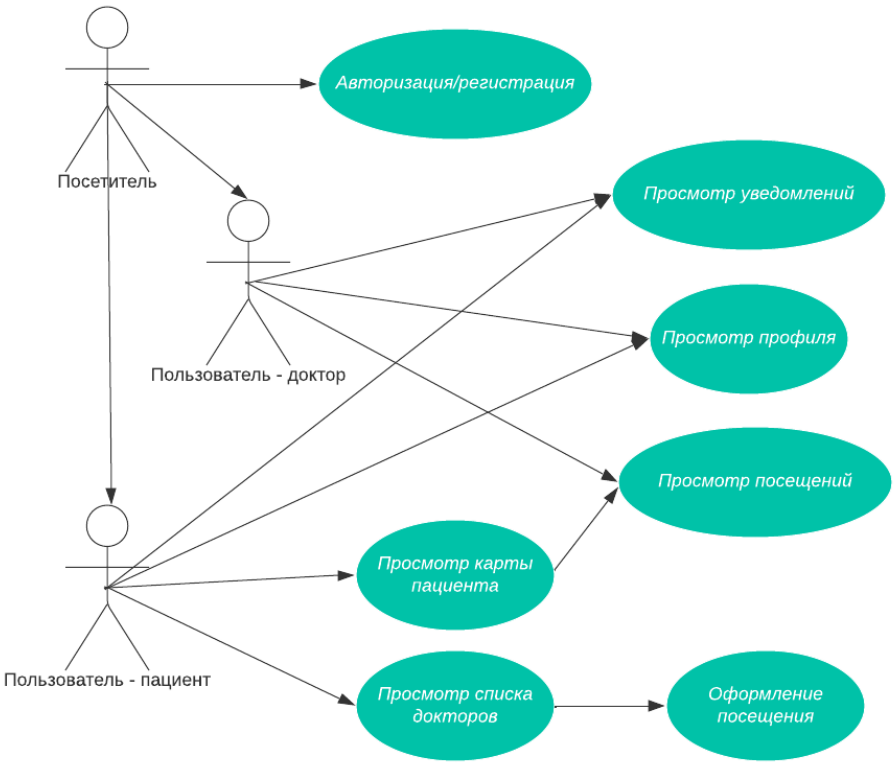


Рисунок 4.1 - Диаграмма вариантов использования

На схеме изображено 3 типа пользователей: анонимный посетитель, доктор, пациент. После авторизации или регистрации у анонимного пользователя появляется роль доктора или пациента. В зависимости от роли пользователям доступен различный функционал интерфейса, тем не менее, у всех типов пользователей имеются несколько одинаковых возможностей, например просмотр и редактирование своего профиля или просмотр уведомлений.

Из некоторых вариантов использования вытекают другие варианты, например из варианта просмотра докторов вытекает вариант оформления посещения для пациента.

В приложении реализована навигационная панель, таким образом пользователи с любой страницы всегда могут перейти на любую другую.

Для примера реализации использования рассмотрим некоторые функции приложения, для реализации которых необходимо применение некоторых нетривиальных алгоритмов:

1. «Быстрый» поиск докторов для пациента;

Доступ к данному поиску имеет пациент, так как доктор в нём не нуждается.

Входными данными в данном случае является строка с ключевыми словами, которые ввёл пациент в строку быстрого поиска.

Выходными данными является список докторов, полученный путём фильтрации списка всех докторов по ключевым словам.

Ключевые слова разделены символом пробела и в бекэнд части происходит разбиение строки на сами ключевые слова, после чего происходит фильтрация списка всех докторов по каждому ключевому слову. Листинг кода, отвечающего за фильтрацию приведён в приложении А. Диаграмма активности выполнения данного кейса приведена ниже (рисунок 4.2).

2. Подтверждение почты для нового пользователя;

После успешной регистрации для входа в систему пользователю необходимо подтвердить почту, путём перехода по ссылке в письме, посланном на почту в момент регистрации нового пользователя.

Входными данными является объект, представляющий собой нового пользователя для регистрации.

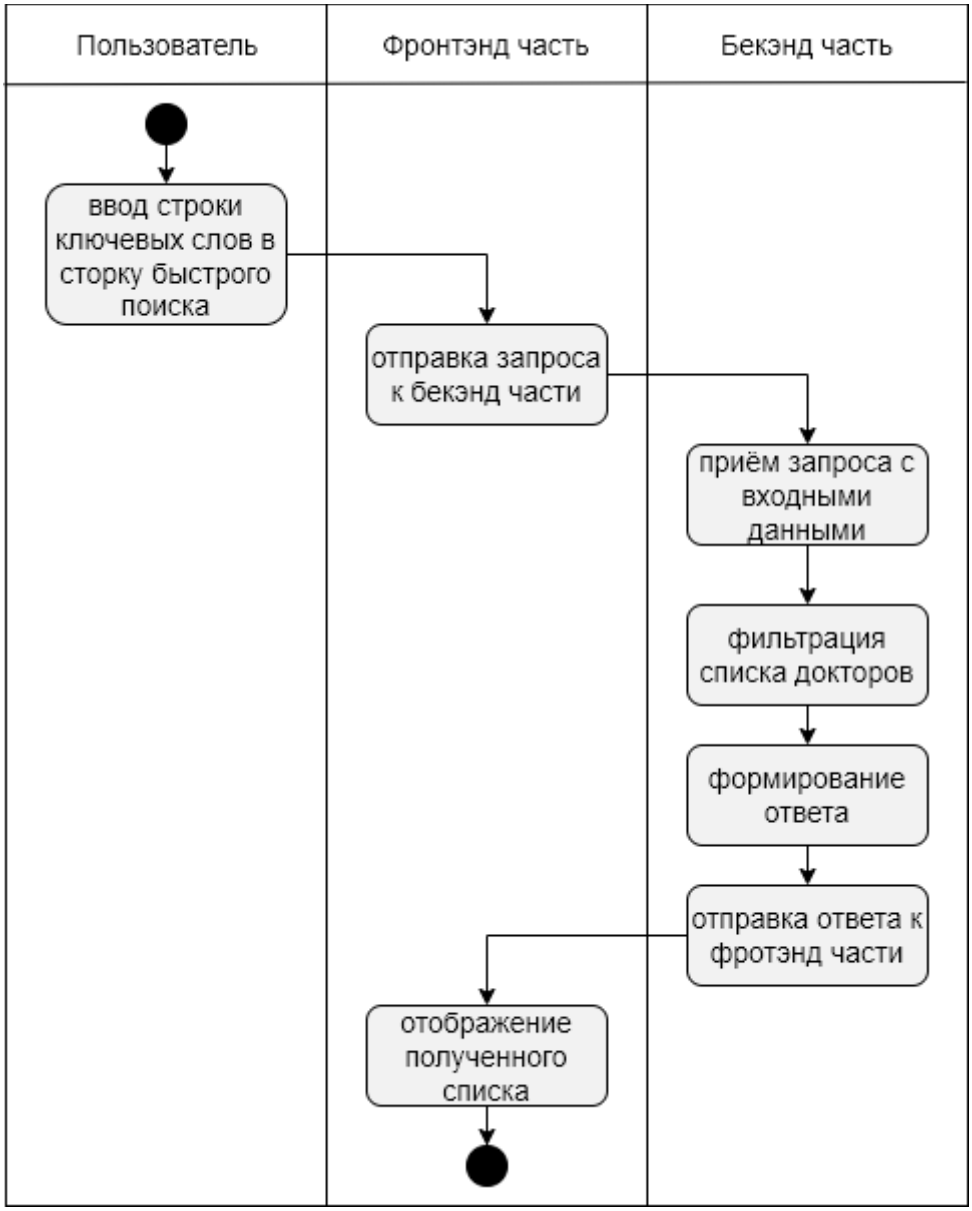


Рисунок 4.2 - Диаграмма активности выполнения кейса «быстрый» поиск докторов

Выходными данными является результат регистрации нового

пользователя, если регистрация прошла неудачно, то результат будет содержать список ошибок, из-за которых это произошло.

При успешной регистрации пользователь переносится на страницу, которая содержит сообщение что необходимо подтвердить почту, перейдя по ссылке в письме, после чего пользователь может успешно войти в систему (рисунок 4.3).

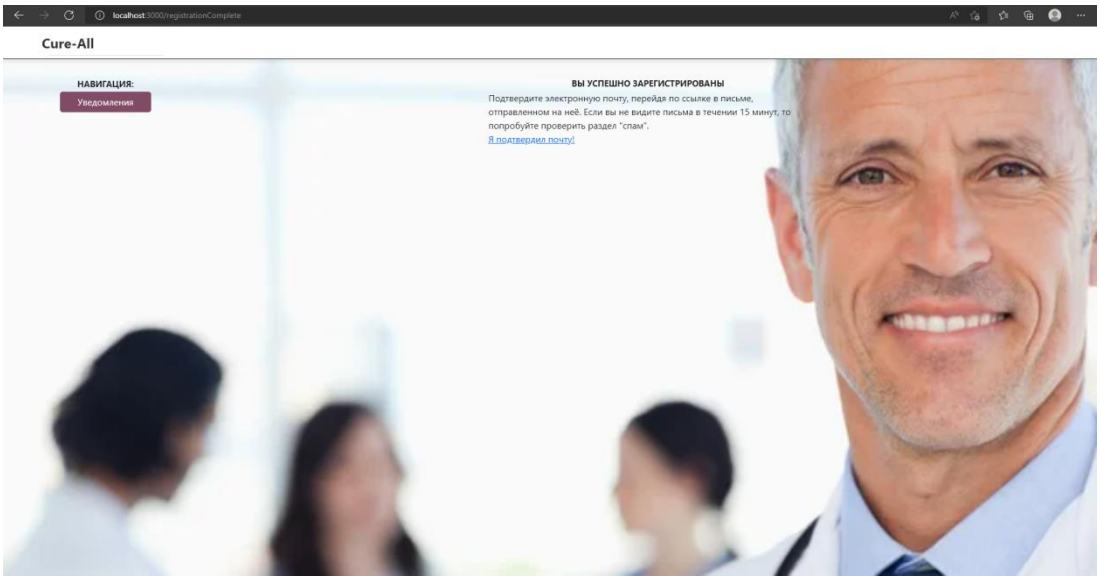


Рисунок 4.3 - Страница, сообщающая о необходимости подтверждения почты

Листинг кода, отвечающего за регистрацию нового пользователя и отправку сообщения для подтверждения почты приведён в приложении А. Диаграмма активности выполнения данного кейса приведена ниже (рисунок 4.4).

3. Получение доступного времени для посещения на дату для доктора;

При регистрации нового запроса на посещения пользователь может выбрать предпочтительную дату, после чего будет выведен список доступного на данную дату времени (рисунок 4.5).

Входными данными являются дата, выбранная пациентом, а также идентификатор доктора в базе данных.

Выходными данными является список доступного на данную дату времени.

Проверка на то, подходит ли определённое время в этот день проходит по многим параметрам, например, учитываются специфические даты доктора(выходной, больничный, отпуск и т. д.), учитывается находится ли данная дата или время в прошлом, учитывается является ли данный день рабочим для доктора и т. д. Листинг кода, отвечающего за вычисление подходящего времени приведён в приложении А.

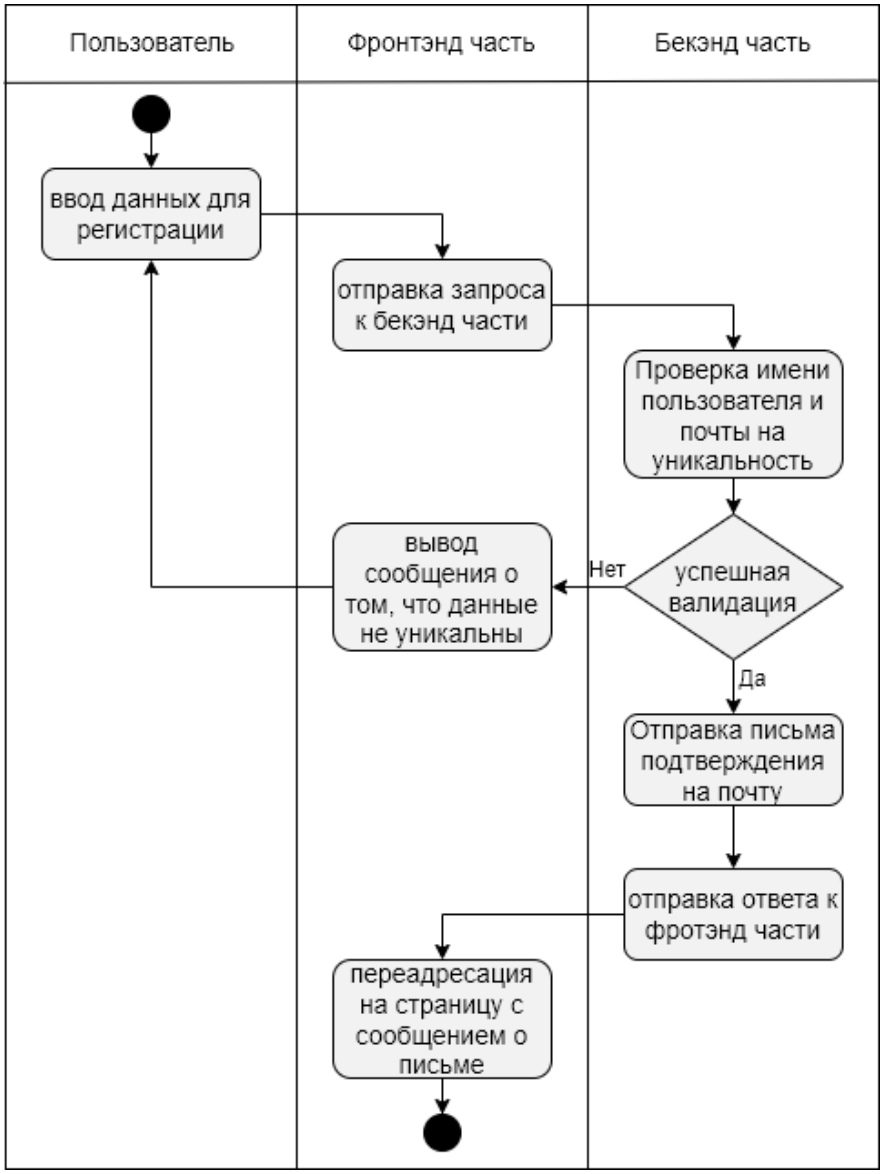


Рисунок 4.4 - Диаграмма активности выполнения кейса подтверждение почты для нового пользователя

Диаграмма активности выполнения данного кейса приведена ниже (рисунок 4.6).

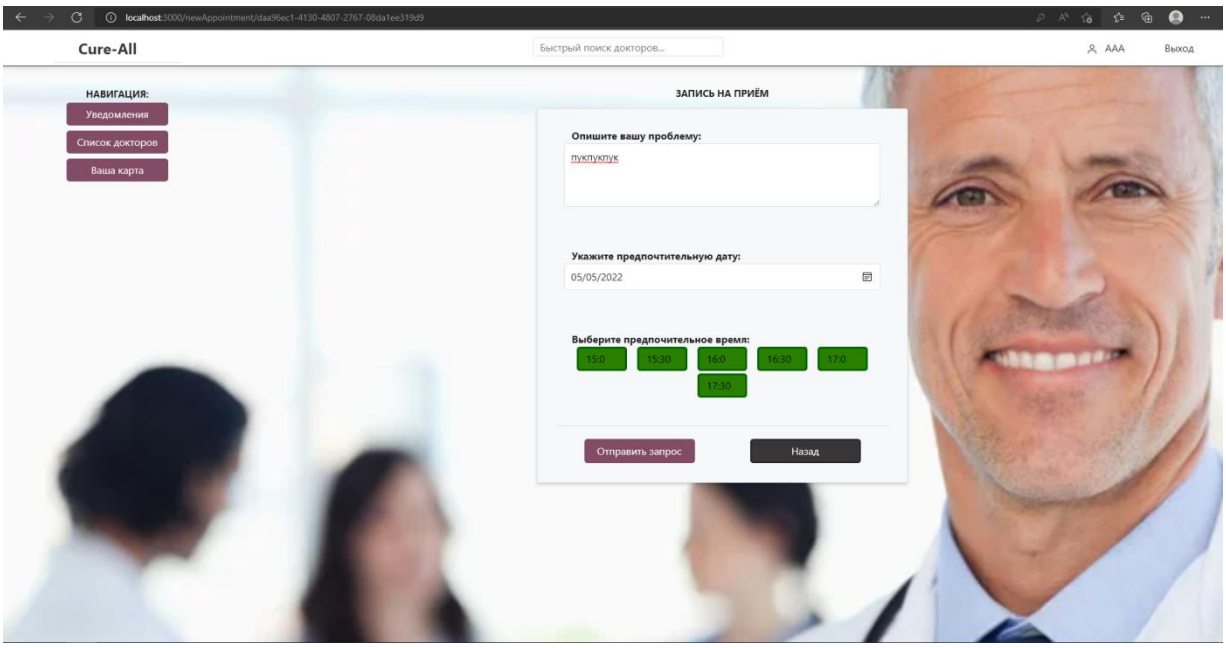


Рисунок 4.5 - Форма оформления посещения, содержащая список доступного на выбранную дату времени

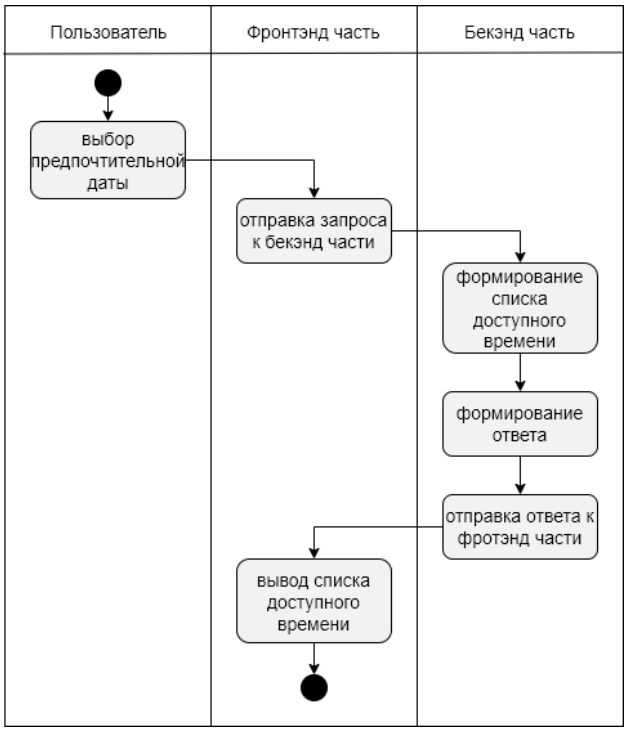


Рисунок 4.6 - Диаграмма активности для выполнения кейса получение доступного времени для посещения на дату для доктора

4.3 Модульное тестирование

Модульное тестирование было проведено для бекэнд части. Для написания методов контроллеров бекэнд приложения была использована библиотека MediatR.

MediatR - это небольшая простая библиотека, которая позволяет обрабатывать сообщения в памяти как команды, применяя декораторы или расширения функциональности. Использование шаблона медиатора помогает уменьшить количество связей и изолировать функциональность, связанную с запрашиваемой работой. При этом вы можете автоматически подключаться к обработчику, который выполняет эту работу — в данном случае к обработчику команд. Ещё одна причина для использования шаблона медиатора была раскрыта Джимми Богардом, создателем данной библиотеки: "Я думаю, здесь стоит упомянуть тестирование — вы получаете ясное и согласованное представление о поведении системы". Запрос поступает, ответ выдаётся — этот принцип оказался очень полезным при разработке согласованно работающих тестов [18].

Таким образом приложение содержит набор команд MediatR’a (рисунок 4.7), которые просто вызываются в методах контроллера, это позволяет не перегружать контроллеры и оставлять их код компактным и легко читаемым. Также это позволяет упростить процесс модульного тестирования, тестировать можно не отдельные методы репозиториев или методы бизнес-логики, а команды MediatR’a и таким образом покрыть тестами большую часть кода программы.

На основе всего вышесказанного были созданы модульные тесты, минимум по одному тесту на практически каждую команду MediatR’a.

Для создания тестов бекэнд части была использована библиотека xUnit.

xUnit - это бесплатный инструмент тестирования с открытым исходным кодом для .NET, который разработчики используют для написания тестов для

своих приложений. По сути, это среда тестирования, которая предоставляет

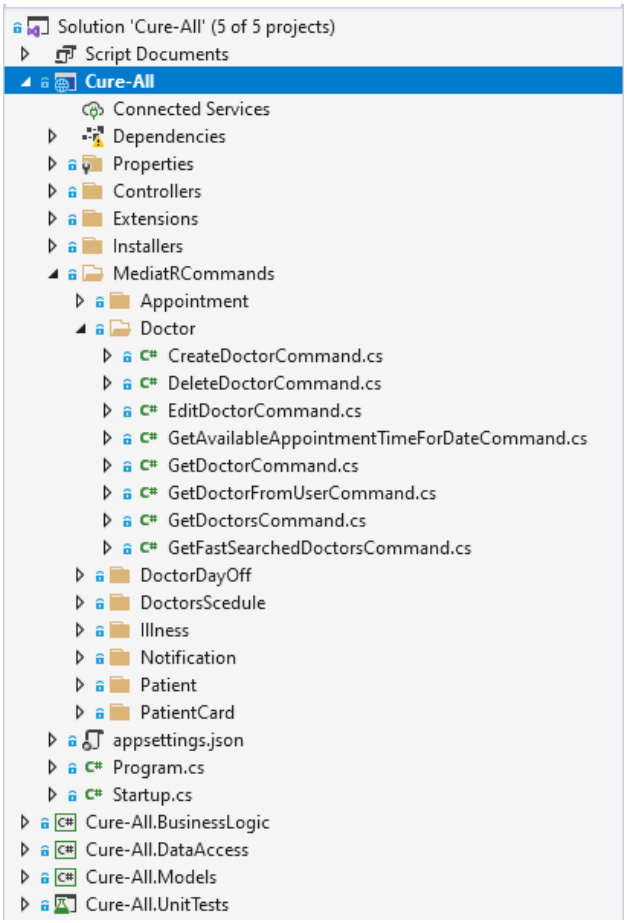


Рисунок 4.7 - Структура расположения команд MediatR'а в проекте

набор атрибутов и методов, которые мы можем использовать для написания тестового кода для наших приложений. Вот некоторые из этих атрибутов, которые можно использовать при работе с xUnit:

- [Fact] - атрибут указывает, что метод должен выполняться исполнителем теста;

- [Theory] - атрибут подразумевает, что мы собираемся отправить некоторые параметры в наш тестовый код. Таким образом, он похож на атрибут [Fact], потому что он утверждает, что метод должен выполняться исполнителем теста, но дополнительно подразумевает, что мы собираемся отправить параметры методу тестирования;

- [InlineData] - атрибут предоставляет те параметры, которые мы отправляем методу тестирования. Если мы используем атрибут [Theory], мы

также должны использовать [InlineData] [19].

Файлы модульных тестов размещаются в отдельном проекте, Cure-All.UnitTests (рисунок 4.8).

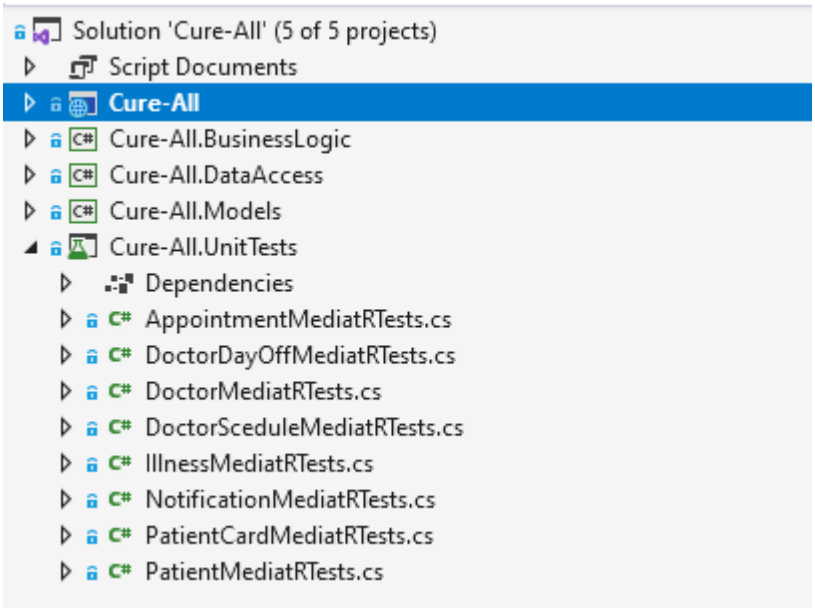


Рисунок 4.8 - Структура проекта, содержащего модульные тесты

Были созданы файлы для каждой из папок в директории MediatRCommands (рисунок 3.7), по одному файлу тестов для каждой папки. Названия файлов тестов формировались по правилу: название папки с командами MediatR'a + MediatRTests, что показывает что в данном файле происходит тестирование команд MediatR'a.

Так как тесты должны выполняться периодически, они не могут затрагивать данные из базы данных, таким образом, для тестирования нужно использовать «заглушку» для методов репозиториев, как будто они действительно возвращают некоторые данные. В качестве такой «заглушки» в проекте выступает фреймворк Moq.

Moq – это простой и легковесный изоляционный фреймворк (Isolation Framework), который построен на основе анонимных методов и деревьев выражений. Для создания моков он использует кодогенерацию, поэтому позволяет «мокать» интерфейсы, виртуальные методы (и даже

защищенные методы) и не позволяет «мокать» невиртуальные и статические методы [20].

Рассмотрим некоторые тесты для ознакомления с ними:

- GetDoctorsCommand_ShouldReturnAllDoctors - тест проверяет команду GetDoctorsCommand, которая должна вернуть всех докторов, которые были заранее указаны при помощи фреймворка Moq. Листинг кода теста приведён в приложении А;

- GetDoctorCommand_ShouldReturnNull - тест проверяет команду GetDoctorCommand, которая должно вернуть пустую ссылку на объект, так как в тесте намеренно указан неверный идентификатор доктора. Листинг кода теста приведён в приложении А;

- CreateDoctorCommand_ShouldCreateNewDoctor - тест проверяет команду CreateDoctorCommand, которая должна создать новый объект и успешно зарегистрировать его путём назначения идентификатора для него. Листинг кода теста приведён в приложении А.

В качестве аргумента в пользу обязательного покрытия кода тестами можно привести пример провала одного из тестов данного проекта, что позволило выявить ошибку.

Тест GetDoctorsCommand_ShouldReturnAllDoctors при первом запуске завершился провалом, причиной послужило то, что в команде MediatR’a, которая тестируется в данном тесте, при написании кода были указаны стандартные параметры фильтрации. Данная проблема была быстро решена после выявления путём очистки стандартных параметров фильтрации.

Результаты выполнения модульных тестов приведены ниже (рисунок 4.9).

Таким образом, написав всего 45 тестов, тестами был покрыт практически весь код бекэнд части. Выполнение всех тестов в итоге завершилось успехом.

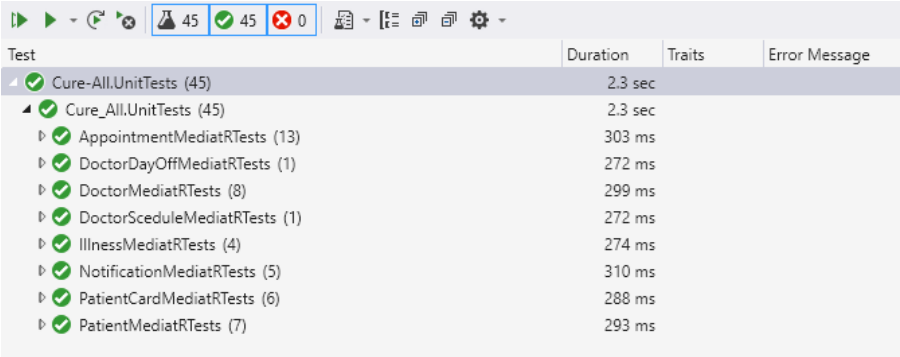


Рисунок 4.9 - Результаты выполнения модульных тестов

4.4 Функциональное тестирование

Функциональное тестирование было проведено над фронтэнд частью так как весь функционал приложения сосредоточен именно здесь. Исходя из того, что на текущий момент теряет актуальность, для функционального тестирования фронтэнд части был использован фреймворк Cypress.

Cypress - это open-source фреймфорк для E2E тестирования. Это также как и Puppeteer относительно молодой инструмент, однако он вносит новые концепции и решения в способы осуществления автоматизации и тестирования. Ключевой особенностью, Cypress является то, что он исполняется внутри самого браузера. Это в том числе означает, что Cypress всегда отслеживает моменты вызова всякого рода событий в браузере и никогда не упустит любые манипуляции с элементами страницы, что намного уменьшает вероятность появления floating тестов.

Достоинства:

- встроенный набор инструментов для тестирования построенный на форке mocha, chai, sinon;

- встроенный механизм автоматического ожидания, это собственно означает, что при написании сценарием нет необходимости писать async/await функции как это делается в Puppeteer и Selenium. Cypress сам подождёт когда появится нужный элемент, подождёт когда

закончится анимация, и подождёт когда очередной сетевой запрос завершится;

- time machine фича, которая позволяет в Cypress test runner откатываться на определённые шаги в последовательности выполнения теста;

- исчерпывающая документация с большим набором примеров;

- возможность написания в том числе и unit тестов [21].

Из-за невозможности предсказать состояние базы данных на момент запуска теста автоматически был протестирован лишь некоторый функционал. Таким
образом были протестированы, например, появление сообщений об ошибке валидации при вводе неверных данных в формы входа в систему и регистрации нового пользователя, а также правильность перехода между страницами при помощи панели навигации и заголовка сайта. Из-за того, что запуск автоматизированных тестов не должен в итоге изменять данные в базе данных стало невозможным автоматически протестировать, например, регистрацию нового пользователя или оформление нового посещения к доктору.

Список созданных для Cypress тестов представлен ниже (рисунок 4.10).

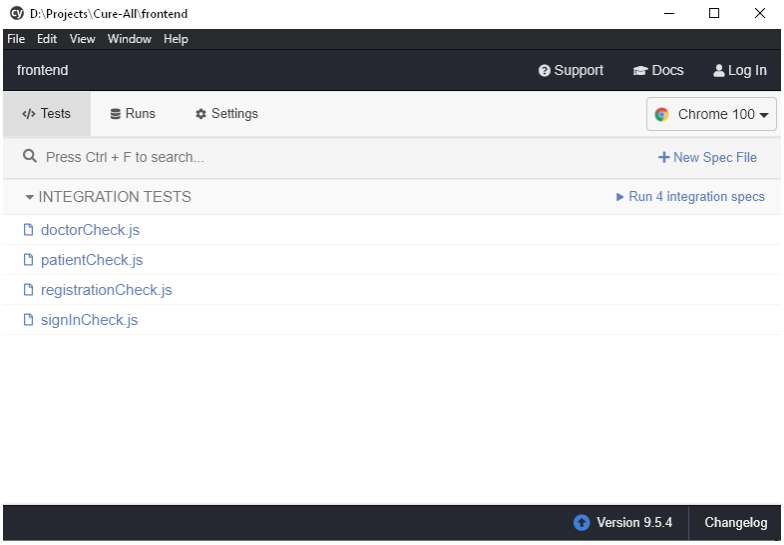
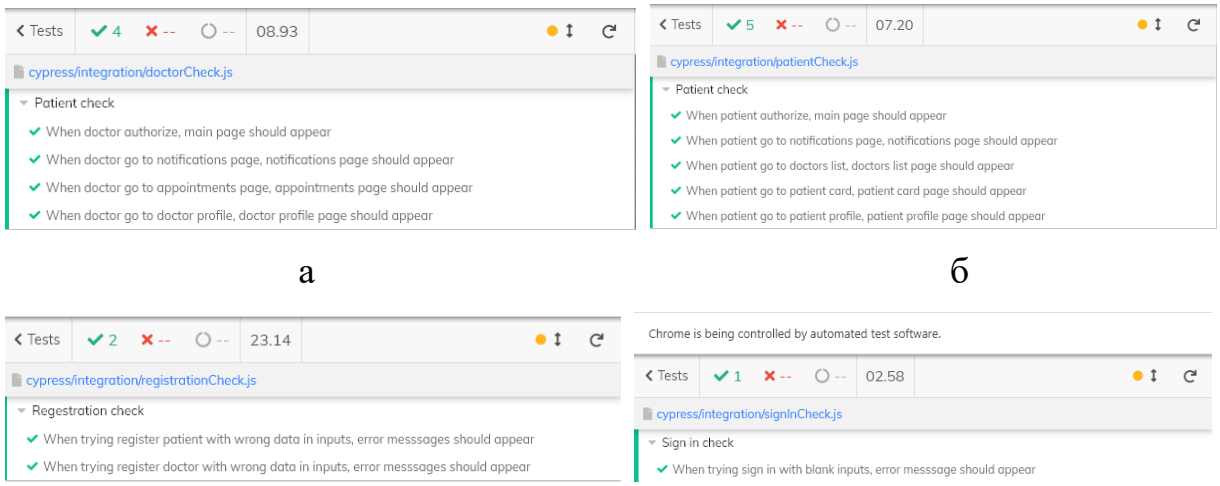


Рисунок 4.10 - Список автоматизированных функциональных тестов

Результаты запуска автоматизированных тестов представлены ниже (рисунок 4.11 а-г).



а б

в г

Рисунок 4.11 - Результаты автоматических тестов: а - результаты теста doctorCheck; б - результаты теста patientCheck; в - результаты теста registrationCheck; г - результаты теста signInCheck

Для ознакомления с кодом функциональных тестов рассмотрим некоторые тесты:

- When doctor go to notifications page, notifications page should appear - тест проверяет что при переходе на страницу со списком уведомлений доктор действительно видит страницу уведомлений. Листинг данного теста приведён в приложении А;

- When trying sign in with blank inputs, error messsage should appear - тест проверяет что при попытке входа в систему с неверными данными формы, появляются ошибки валидации. Листинг данного теста приведён в приложении А;

- When trying register doctor with wrong data in inputs, error messsages should appear - тест проверяет что при попытке регистрации нового доктора используя неверные данные будут появляться ошибки валидации. Листинг

данного теста приведён в приложении А.

Остальной функционал приложения был протестирован вручную. Чек-лист тестирования приведён ниже (таблица 4.1).

Таблица 4.1 - Чек-лист тестирования ИС «Веб-сервис для записи к врачу»

Тестируемый модуль	**ИД кейса**	**Тестируемое требование**	**Результат**
Аутентификация	ТС-1	Регистрация нового доктора	Выполнено успешно
	ТС-2	Регистрация нового пациента	Выполнено успешно
	ТС-3	Вход в систему под различными пользователями	Выполнено успешно
	ТС-4	Редактирование профиля пользователя	Выполнено успешно
	ТС-5	Удаление профиля пользователя	Выполнено успешно
Посещения	ТС-6	Изменение посещения доктором	Выполнено успешно
	ТС-6	Оформление нового посещения к доктору	Выполнено успешно
	ТС-8	Отклонение нового посещения доктором	Выполнено успешно
	ТС-9	Принятие нового посещения доктором	Выполнено успешно
Список докторов	ТС-10	Фильтрация по имени	Выполнено успешно
	ТС-11	Фильтрация по специализации	Выполнено успешно
	ТС-12	Фильтрация по городу	Выполнено успешно
	ТС-13	Фильтрация по стране	Выполнено успешно
	ТС-14	Фильтрация по опыту	Выполнено успешно
	ТС-15	Сортировка по опыту	Выполнено успешно

Ниже представлена группа тест-кейсов (таблица 4.2), содержащая по одному тест-кейсу из каждого модуля приведённого выше чек-листа (таблица 4.1).

Таблица 4.2 - Тест-кейсы тестирования ИС «Веб-сервис для записи к врачу»

ИД кейса	**Название**	**№ шага**	**Действие**	**Данные**	**Ожидаемый результат**
1	**2**	**3**	**4**	**5**	**6**
ТС-2	Регистрация нового пациента	1	Переход на страницу регистрации		
		2	Выбор кнопки «Пациент»		
		3	Ввод данных для регистрации	Имя - Пациент; Фамилия - Пациент; Имя пользователя - NewTestPatient; Эл. почта - newTestPatient@gmail.com; Дата рождения - 06.05.2022; Номер телефона - +375296105401; Почтовый код - 12345; Страна - Беларусь; Город - Витебск; Пароль - NewPassword123!; Подтверждение пароля - NewPassword123!.	

Продолжение таблицы 4.2

1	2	3	4	5	6
		4	Нажатие на кнопку подтверждения		Переход на страницу с сообщением о том, что необходимо подтвердить почту
		5	Переход по ссылке из письма на почте		Переход на страницу с успешным подтверждением почты
		6	Попытка входа в систему		Успешный переход на главную страницу
ТС-6	Оформление нового посещения к доктору	1	Вход в систему под любым пациентом		
		2	Переход к списку докторов		
		3	Выбор любого доктора из списка		
		4	Нажатие на кнопку «Записаться на приём»		

Окончание таблицы 4.2

1	2	3	4	5	6
		5	Заполнение формы для нового посещения	Проблема - Тест проблема; Предпочтительноя дата - 06.05.2022; Предпочтительное время - 12:00.	
		6	Нажатие на кнопку подтверждения		Успешное оформление посещения
ТС-11	Фильтрация по специализации	1	Вход в систему под любым пациентом		
		2	Переход к списку докторов		
		3	Выбор специализации для фильтрации списка	Специализация - Невролог	
		4	Нажатие на кнопку подтверждения		Получение списка докторов с выбранной специализацией

Таким образом тестами был покрыт практически весь функционал приложения. Все тесты в итоге завершились успешно.

4.5 Прочие виды тестирования ПС

4.5.1 Тестирование безопасности

Тестирование безопасности - это тип тестирования программного обеспечения, который выявляет уязвимости, угрозы, риски в программном приложении и предотвращает атаки от злоумышленников. Целью тестов безопасности является выявление всех возможных лазеек и слабых мест в программной системе, которые могут привести к потере информации, доходов, репутации со стороны сотрудников или посторонних лиц Организации. Целью тестирования безопасности является выявление угроз в системе и оценка её потенциальных уязвимостей, чтобы система не перестала функционировать или использовалась. Это также помогает в обнаружении всех возможных угроз безопасности в системе и помогает разработчикам в устранении этих проблем посредством кодирования [22].

За регистрацию и авторизацию пользователей в системе отвечает ASP.NET Identity, поэтому рассмотрим вопросы безопасности данной системы.

Платформа ASP.NET Identity полностью интегрирована в ASP.NET. Это означает, что вы можете использовать стандартные средства аутентификации/авторизации MVC вместе с Identity, такие как атрибут Authorize. Атрибут Authorize по умолчанию ограничивает доступ к методам действий для пользователей, не прошедших аутентификацию. Пользователь может зарегистрировать новый аккаунт, информация о котором будет, чаще всего, храниться в базе данных, в данном случае MS SQL Server. ASP.NET Identity поддерживает двухфакторную аутентификацию, когда пользователь для аутентификации должен ввести что-то дополнительное. Наиболее распространёнными примерами является ввод закрытого токена Secureid или код проверки подлинности, который может быть отправлен на email-адрес или по СМС [23].

По умолчанию в ASP.NET Identity невозможно зарегистрировать двух пользователей, использующих одинаковые имена пользователя или адреса

почты, что предотвращает огромное количество ошибок, как, например, невозможность определить то, какой именно пользователь пытается войти в систему из-за того, что логин, являющийся именем пользователя или адресом электронной почты, присвоен нескольким пользователям.

Также для обеспечения дополнительной безопасности в приложении реализована система подтверждения почты, без подтверждённой почты пользователь не сможет войти в систему. Также реализована система смены пароля пользователя если он его забыл, письмо с ссылкой на страницу смены пароля приходит на почту пользователя.

4.5.2 Нагрузочное тестирование

Результаты нагрузочного тестирования приложения в большей степени зависят от того, насколько быстро осуществляются запросы к базе данных. В данном приложении в роли СУБД выступает Microsoft SQL Server, таким образом можно рассмотреть результаты мониторинга работы базы данных (рисунок 4.12).

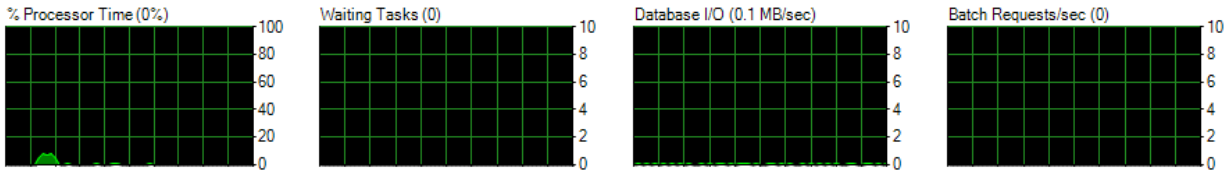


Рисунок 4.12 - Результаты мониторинга работы базы данных

На рисунке 4.12 представлены результаты работы одного пользователя с БД, бесплатная версия Microsoft SQL Server не позволяет смоделировать ситуацию работы с БД нескольких человек, но, можно предположить, что нагрузочные результаты будут расти примерно с той же скоростью с которой будет расти количество пользователей.

4.5.3 Объёмное тестирование

Для объёмного тестирования в базу данных были добавлены 100 докторов, результаты мониторинга работы базы данных при увеличенном количестве данных приведены ниже (рисунок 4.13).

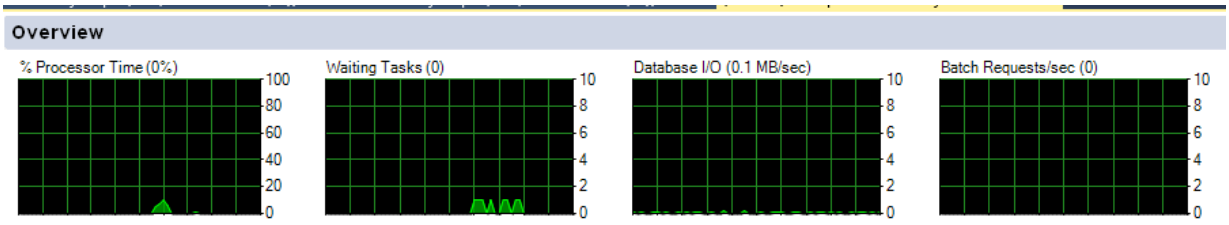


Рисунок 4.13 - Результаты мониторинга работы базы данных с увеличенным количеством данных

Таким образом, сравнивая результаты работы с базой данных с небольшим количеством данных (рисунок 4.12) с увеличенным количеством данных (рисунок 4.13), можно заметить что появляются задачи в режиме ожидания, то есть, увеличивается время отклика на запрос.

5 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Экономическая целесообразность разработки и внедрения программного обеспечения определяется экономическим эффектом, который будет получен производителями при их реализации и потребителями при их использовании. По величине ожидаемого экономического эффекта принимается решение о целесообразности инвестиций в разработку того или иного программного продукта. По характеру объекта вложений инвестиции в разработку программного обеспечения относят к интеллектуальным инвестициям.

При создании программного продукта важно оценить его себестоимости (затраты на разработку).

5.1 Расчёт общей трудоёмкости разработки программного обеспечения и трудоёмкости отдельных стадий разработки

Затраты времени на разработку ПО определяются эмпирическим путём, то есть при помощи органов чувств, в частности, путём наблюдения или эксперимента. Затраты времени включают:

- затраты труда на подготовку и описание задачи - $t_{оп}$;
- затраты труда на исследование алгоритма решения задачи - $t_{ис}$;
- затраты труда на разработку алгоритма (блок-схем) - $t_{ал}$;
- затраты труда на программирование алгоритма по блок-схеме - $t_{пр}$;
- затраты труда на отладку программы - $t_{опр}$;
- затраты труда на подготовку документов по задаче состоят из затрат труда на подготовку рукописей и времени на оформление документов - $t_{д}$.

Суммарные затраты труда рассчитываются как сумма составных затрат труда по формуле (5.1):

$$\sum t = t_{оп} + t_{ис} + t_{ал} + t_{пр} + t_{отл} + t_{д} \quad (5.1)$$

Расчёт суммарных затрат времени представлен в таблице 5.1

Таблица 5.1 - Ориентировочное распределение затрат времени

Вид работ	Трудоёмкость в часах	
	всего, человеко-часов	в том числе машинное время
Подготовка и описание задачи	16	–
Исследование алгоритма решения задачи	16	–
Разработка алгоритма	24	–
Программирование алгоритма	224	224
Отладка программы	24	24
Подготовка и оформление документов	32	32
Итого:	$\Sigma t = 336$	$\Sigma t_{маш} = 280$

Таким образом, суммарные затраты труда на разработку программного обеспечения составили 336 человеко-часов, а также 280 часов машинного времени.

5.2 Расчёт сметы затрат на разработку программного обеспечения

Затраты на оплату ($З_{от}$) труда разработчика ПО включают затраты на оплату труда и отчисления от фонда заработной платы.

Затраты на оплату труда разработчика ПО рассчитываются в бел. руб. по формуле (5.2):

$$З_{ОТ} = \frac{Змес \times \sum t}{КЧР} \quad (5.2)$$

где:

$З_{мес}$ - месячная заработная плата инженера-программиста, руб.;

КЧР - среднемесячная расчётная норма рабочего времени (среднее количество часов работы в месяц в 2022 году при пятидневной рабочей неделе составляет 170,4 часа), час;

$\sum t$ - суммарные затраты труда на разработку и сопровождение ПО (таблица 6.1), ч.;

Месячная заработная платы инженера-программиста включает:

а) оклад;

б) стимулирующие выплаты (надбавки и премии);

в) компенсирующие выплаты (доплаты), которые не учитываются при расчёте заработной платы в условиях дипломного проекта.

Оклад рассчитывается по формуле (5.3):

$$О_к = БС \times ТК \quad (5.3)$$

где:

БС - базовая ставка работников бюджетных организаций, руб.;

ТК - тарифный коэффициент, соответствующий разряду работ разработчика ПО.

С 1 января 2020 г. в соответствии с постановлением Совета Министров Республики Беларусь от 28.02.2019 № 138 «Об оплате труда работников бюджетных организаций» расчёт оплаты труда инженера-программиста, работающего в бюджетной организации, производится исходя из 4 разряда работ (тарифный коэффициент составляет 1,21). Базовая ставка в 2022 году составляет 180 руб.

$$О_к = 180 \times 1{,}21 = 217{,}8 \text{ руб.}$$

Стимулирующие выплаты:

1) Надбавка за работу в бюджетной организации (70% от оклада):

$$Н_б = 0,70 \times 217,8 = 152,46 \text{ руб.}$$

2) Надбавка за стаж работы в бюджетной организации при стаже работы до 5 лет устанавливается в размере 10% от базовой ставки:

$$Н_с = 0,10 \times 180 = 18 \text{ руб.}$$

3) Надбавка за контрактную форму найма (19% оклада):

$$Н_к = 0,19 \times 217,8 = 41,38 \text{ руб.}$$

4) Премия ежемесячная (5% от оклада):

$$П_р = 0,05 \times 217,8 = 10,89 \text{ руб.}$$

Таким образом, заработная плата в месяц определяется по формуле (5.4):

$$З_{мес} = О_к + Н_б + Н_с + Н_к + П_р \qquad (5.4)$$

где:

Ок - оклад работника, руб.;

$Н_б$ - надбавка за работу в бюджетной организации, руб.;

$Н_с$ - надбавка за стаж работы в бюджетной организации, руб.;

$Н_к$ - надбавка за контрактную форму найма, руб.;

$П_р$ – ежемесячная премия, руб.

$$З_{мес} = 217,8 + 152,46 + 18 + 41,38 + 10,89 = 440,53 \text{ руб.}$$

Таким образом, можем рассчитать оплату труда разработчика ПО по формуле (5.2):

$$З_{ОТ} = \frac{440,53 \times 336}{170,4} = 868,65(руб.)$$

Отчисления от фонда оплаты труда рассчитываются по формуле (5.5):

$$O_{Зот} = \frac{O_{ФСЗН} + O_c}{100} \times З_{ОТ}, \quad (6.5)$$

где:

$О_{фсзн}$ - отчисления в Фонд социальной защиты населения (ставка отчислений составляет 34% от всех выплат работнику), руб.;

$О_с$ - страхование нанимателя от несчастных случаев на производстве и профзаболеваний (ставка отчислений составляет 0,6% от всех выплат работнику), руб.

$$O_{Зот} = \frac{34 + 0,6}{100} \times 868,65 = 300,55(руб.)$$

Затраты на оплату труда с учётом отчислений рассчитываются по формуле (5.6):

$$ФЗП = З_{от} + О_{Зот} \quad (5.6)$$

$$ФЗП = 868,65 + 300,55 = 1169,2 \text{ (руб.)}$$

5.3 Эксплуатационные затраты на оборудование

Стоимость оборудования не включается в себестоимость разработки программного обеспечения, но само оборудование используется при разработке ПО. При создании программного обеспечения в качестве оборудования используется персональный компьютер, стоимость которого составляет:

$$С_{обор} = 1750,00 \text{ (руб.)}$$

Суммарная годовая стоимость эксплуатационных затрат $С_э$ рассчитывается по формуле (5.7):

$$C_Э = C_{ТО} + C_{ЭЭ} + A_{год}, \quad (6.6)$$

где:

$С_{то}$ - затраты на техническое обслуживание и ремонт оборудования, руб.;

$С_{ээ}$ - годовая стоимость электроэнергии, руб.;

$А_{год}$ - годовые амортизационные отчисления, руб.

Затраты на техническое обслуживание и ремонт составляют 3% от стоимости оборудования:

$$С_{то} = 0{,}03 \times 1750 = 52{,}5 \text{ (руб.)}$$

Амортизационные отчисления, процесс постепенного переноса стоимости основных средств производства на себестоимость продукции (по мере их материального износа или морального устаревания). Амортизационные отчисления производятся по установленным нормам амортизации, выражаются в процентах к стоимости оборудования и рассчитываются по формуле (5.7):

$$A_{год} = C_{обор} \cdot \frac{H_A}{100\%}, \quad (5.7)$$

где:

$С_{обор}$ - стоимость персонального компьютера;

$Н_А$ - норма амортизации, которая рассчитывается по формуле (5.8):

$$H_A = \frac{100\%}{T_{норм}}, \quad (5.8)$$

где:

$T_{норм}$ - нормативный срок службы (для персонального компьютера 5 лет).

$$H_A = \frac{100}{5} = 20\%$$

$$A_{год} = 1750 \times \frac{20}{100} = 350(руб.)$$

Стоимость электроэнергии вычисляется по формуле (5.9):

$$C_{ЭЭ} = M \times k_з \times F_{эф} \times C_{кВт \times ч} \times K_C, \quad (5.9)$$

где:

M - мощность компьютера, равная 0,1 КВт;

$k_з$ - коэффициент загрузки, учитывающий использование оборудования по времени (0,8);

$C_{кВт×ч}$ - стоимость 1 кВт-час электроэнергии (0,27553 руб./кВтч для бюджетных организаций по состоянию на май 2022 года);

$K_с$ - коэффициент, учитывающий потери в сети (1,05);

$F_{эф}$ - эффективный фонд рабочего времени, рассчитывающийся по формуле (5.10):

$$F_{эф} = Д_{ном} \cdot d \cdot \left(1 - \frac{f}{100\%}\right), \quad (5.10)$$

где:

$Д_{ном}$ = 255 - номинальное число рабочих дней в 2022 году при пятидневной рабочей неделе;

d = 7,988 - средняя продолжительность рабочего дня в 2022 году, час.;

f = 2% - планируемый процент времени на ремонт оборудования.

$$F_{эф} = 255 \times 7{,}988 \times (1 - 0{,}02) = 1996{,}2 \text{ (час.)};$$

$$C_{ээ} = 0{,}1 \times 0{,}8 \times 1996{,}2 \times 0{,}27553 \times 1{,}05 = 46{,}2 \text{ (руб.)}$$

Наконец, рассчитав затраты на техническое обслуживание и ремонт, амортизационные отчисления и годовую стоимость электроэнергии можем рассчитать суммарную годовую стоимость эксплуатационных затрат:

$$C_{э} = 52{,}5 + 350 + 46{,}2 = 128{,}7 \text{ (руб.)}$$

Однако, данная стоимость эксплуатационных затрат рассчитана да весь год. Необходимо скорректировать полученное значение с учётом временного коэффициента (так как оборудование будет эксплуатироваться не весь год, а только в течение времени $\Sigma t_{маш}$). Корректировка проводится по формуле (5.11):

$$Э_{з} = \sum t_{маш} \cdot \frac{C_{э}}{F_{эф}} [руб], \qquad (5.11)$$

где:

$C_{э}$ - суммарная годовая стоимость эксплуатационных затрат;

$F_{эф}$ - эффективный фонд рабочего времени, час.;

$\Sigma t_{маш}$ - общее время использования оборудования (таблица 6.1).

$$Эз = 280 \times \frac{128{,}7}{1996{,}2} = 18{,}05 (руб.)$$

5.4 Затраты на материалы

Затраты на материалы состоят из расходов на бумагу, канцелярские принадлежности и прочие материалы, которые необходимы в процессе разработки ПО. Расчёт стоимости затрат на материалы производится по нормативу (Н) на 100 строк кода (принимается 1,20 руб.), с учётом общего объёма команд (V_K), который определяется по факту, исходя из количества команд при разработке ПО. Расчёт производится по формуле (5.12):

$$З_М = \frac{V_К \times Н}{100}, \quad (6.14)$$

где:

V_K - общий объём команд в коде программы;

Н - норматив затрат на материалы в расчёте на 100 команд, руб.

Также, стоимость затрат на материалы, необходимые для разработки ПО, можно рассчитать по фактическим, понесённым затратам, которые составили 40 рублей.

5.5 Накладные расходы

Накладные расходы, связанные с управлением, организационными расходами и прочими дополнительными затратами составляют 50% от оплаты труда разработчика ПО:

$$С_{накл} = 0,5 \times 868,65 = 434,33 \text{ (руб.)}$$

5.6 Смета затрат на разработку программного обеспечения

Суммарные затраты на разработку ПО считаются как сумма фонда заработной платы и отчислений от него, эксплуатационных затрат, затрат на

материалы, накладных расходов.

Расчёт стоимости разработки ПО представлен ниже (таблица 6.2):

Таблица 6.2 - Смета затрат на разработку программного обеспечения

Наименование статей затрат	Обозначение	Сумма, руб.
1	2	3
1. Заработная плата	$З_{от}$	868,65
2. Отчисления от заработной платы	$О_{Зот}$	300,55
3. Эксплуатационные расходы по оборудованию	$Э_{з}$	18,05
4. Затраты на материалы	$З_{м}$	40
5. Накладные расходы	$С_{накл}$	434,33
Итого затраты на разработку	$С_{полн}$	1661,58

5.7 Расчёт экономического эффекта от разработки программного обеспечения (для разработчика ПО)

Заказчик оплачивает разработчику всю сумму расходов по проекту (итоговые затраты на разработку по таблице 6.2) с учётом прибыли разработчика и налога на добавленную стоимость с учётом качества, потребительских свойств продукции (ПО) и конъюнктуры рынка. Таким образом, отпускная цена программного обеспечения представляет собой не цену за единицу продукции, а цену проекта вместе с его исходным кодом и документацией, за которую его можно продать и получить определённую выгоду. Прогнозируемая отпускная цена ПО ($Ц_{по}$) с учётом НДС рассчитывается по формуле (5.15):

$$Ц_{по} = \frac{(C_{полн} + П) \times (100 + СТ_{НДС})}{100}, \quad (5.15)$$

где:

$С_{полн}$ - полная (плановая) себестоимость ПО, руб.;

П - прибыль разработчика ПО, руб.;

$СТ_{НДС}$ - ставка налога на добавленную стоимость (20%), в %.

Прибыль закладывается в цену исходя из уровня рентабельности, рассчитывается по формуле (5.16):

$$П = \frac{R \times C_{полн}}{100}, \qquad (5.16)$$

где:

R - уровень рентабельности (20%), в %;

$С_{полн}$ - плановая себестоимость (таблица 6.2), руб.

Таким образом, прибыль составляет:

$$П = \frac{20 \times 1661{,}58}{100} = 332{,}32(руб.)$$

С учётом вычисленной прибыли, прогнозируемая цена ПО ($Ц_{по}$) с учётом НДС составит:

$$Ц_{ПО} = \frac{(1661{,}58 + 332{,}32) \times (100 + 20)}{100} = 2392{,}68(руб.)$$

Имея ввиду то, что программное обеспечение разрабатывается для одного объекта, в качестве экономического эффекта разработчика от реализованного программного обеспечения можно рассматривать чистую прибыль (ЧП), которая рассчитывается по формуле (5.17):

$$ЧП = \frac{П \times (100 - СТ_{П})}{100}, \quad (6.17)$$

где:

$СТ_{п}$ - ставка налогообложения прибыли (в 2022 году составляет 18%), в %.

$$ЧП = \frac{332{,}32 \times (100 - 18)}{100} = 272{,}50(руб.)$$

Таким образом, разработчик программного обеспечения может путём продажи созданного ПО заказчику по отпускной цене 2392,68 рублей, что покроет затраты на разработку ПО в размере 1661,58 рублей, получить чистую прибыль равную 272,50 рубля.

6 ОХРАНА ТРУДА

Согласно Закону об охране труда от 23 июня 2008 г. № 356 – З(в ред. Закона Республики Беларусь от 12.07.2013 N 61-З) даётся следующее определение понятию охраны труда:

Охрана труда – система обеспечения безопасности жизни и здоровья работников в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства[24].

Систему законов, регулирующих вопросы охраны труда в Республике Беларусь составляют: Конституция Республики Беларусь, Концепция государственного управления охраной труда Республики Беларусь, Трудовой кодекс Республики Беларусь, Законы Республики Беларусь «Об охране труда», «Об основах государственного социального страхования», «О пенсионном обеспечении», «О санитарно-эпидемическом благополучии населения», «О техническом нормировании и стандартизации», «О пожарной безопасности», «О промышленной безопасности», «О радиационной безопасности населения», «О защите населения и территорий от чрезвычайных ситуаций природного и техногенного характера», «О здравоохранении», «О предприятиях» и др.,ТКП, НПА, ТНПА, ЛНПА.

Охрана труда имеет большое социальное и экономическое значение.

Социальное значение охраны труда:

- сохранение работоспособности и трудового долголетия работника;

- охрана жизни и здоровья работника от возможных воздействий вредных условий производства;

- способствование гуманизации труда, содействие его культурно-техническому росту.

Экономическое значение охраны труда:

- рост производительности труда работников, производства и экономики;

- экономия фонда социального страхования и сокращение потерь рабочего времени.

Работа с разработанным в данном дипломном проекте веб-сервисом предполагает нахождение в жилом помещении, которое не нуждается в реализации каких-либо специальных требований для охраны труда. Специальной службы по охране труда не предусмотрено.

Характеристика объекта с точки зрения охраны труда будет рассмотрена на примере администратора разработанного веб-сервиса.

Проведём оценку факторов производственной среды, тяжести и напряжённости трудового процесса (таблицы 6.1 - 6.3).

Таблица 6.1 - Оценка факторов производственной среды

Факторы и показатели производственной среды	Гигиенические нормативы (ПДК, ПДУ)	Фактические величины
1	2	3
Шум, дБА, дБ	60	40
Электромагнитные поля и неионизирующие излучения		
Напряженность электрического поля, В/м		
– от 5 Гц до 2 кГц	25	21
– от 2 кГц до 400 кГц	2,5	0,6
Плотность магнитного потока, нТл		
– от 5 Гц до 2 кГц	250	210
– от 2 кГц до 400 кГц	25	6
Электростатические поля, кВт/м	15	5,6

Окончание таблицы 6.1

1	2	3
Микроклимат		
– Температура воздуха, ºС	18-24	20
– Относительная влажность, %	не более 60	50
– Скорость движения воздуха, м/с	не более 0,3	0,1
Освещённость, лк	300	560

Таблица 6.2 - Оценка тяжести трудового процесса

Показатели тяжести трудового процесса	Фактическое значение показателя
Физическая динамическая нагрузка, кгм	
– Региональная нагрузка при перемещении груза на расстояние до 1 м	До 2500
– Общая нагрузка при перемещении груза на расстояние от 1 до 5 м	До 12500
Масса поднимаемого и перемещаемого груза вручную, кг	До 2
– Подъём и перемещение тяжести при чередовании с другой работой	3-12,5
– Подъём и перемещение тяжестей постоянно в течение рабочей смены	10
– Суммарная масса грузов, перемещаемых в течение каждого часа смены с рабочей поверхности	До 350
Стереотипные рабочие движения, количество за смену	
– При локальной нагрузке	12 000
– При региональной нагрузке	
Статическая нагрузка, кг (силы) · с	
– Одной рукой	До 36 000
– Двумя руками	20 000
– С участием мышц корпуса, ног	
Рабочая поза (стоя)	Стоя 20 %
Наклоны корпуса	10
Перемещения в пространстве, обусловленные технологическим процессом, км	
– По горизонтали	До 4
– По вертикали	

Таблица 6.3 - Оценка напряжённости трудового процесса

Показатели напряжённости трудового процесса	Характеристика показателей в соответствии с гигиеническими критериями
1	2
Интеллектуальные нагрузки	
1 Содержание работы	Решение задач по инструкции
2 Восприятие сигналов (информации) и их оценка	Восприятие сигналов, но не требуется коррекция действий
3 Распределение функций по степени сложности задания	Обработка и выполнение задания
4 Характер выполняемой работы	Работа по установленному регламенту
Сенсорные нагрузки	
1 Длительность сосредоточенного наблюдения (в % от времени смены)	До 25
2 Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы	60
3 Число производственных объектов одновременного наблюдения	1
4 Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0,5 м) в мм при длительности сосредоточенного наблюдения (% времени смены)	0,3-0,5 мм-до 30% более 0,5 мм-до 70%
5 Наблюдение за экранами видеотерминалов (часов в смену):	
- при буквенно-цифровом типе отображения информации;	5
- при графическом типе отображения	До 3
6 Нагрузка на слуховой анализатор (при производственной необходимости восприятия речи или дифференцированных сигналов)	Разборчивость слов и сигналов от 75% до 50%. Помехи присутствуют

Окончание таблицы 6.3

1	2
7 Нагрузка на голосовой аппарат (суммарное количество часов, наговариваемое в неделю)	3 - 7
Эмоциональные нагрузки	
1 Степень ответственности за результат собственной деятельности. Значимость ошибок	Ответственность за качество работы, влечёт дополнительные усилия со стороны руководства
2 Степень риска для собственной жизни	Исключена
3 Степень ответственности за безопасность других лиц	Исключена
Монотонность нагрузок	
1 Число элементов (приемов), необходимых для реализации простого задания или в многократно повторяющихся операциях	8
2 Продолжительность выполнения простых производственных заданий или повторяющихся операций, с	25-100
3 Монотонность производственной обстановки (время пассивного наблюдения за ходом техпроцесса в % от времени смены)	76-80
Режим работы	
1 Сменность работы	Односменная

Из данных таблиц 6.2 - 6.3 можно сделать вывод, что администратор веб-сервиса подвержен некоторым не физическим видам нагрузки, в данном случае: Эмоциональной, сенсорной и интеллектуальной.

Таким образом, разработаем карту рисков рабочего места администратора.

Для оценки рисков применяем классический метод. Оценка рисков рассчитывается по формуле (6.1):

$$R = P \times S, \quad (6.1)$$

где:

R - риск, балл;

P - вероятность возникновения опасности, балл;

S - серьёзность последствий воздействия опасности, балл.

Путём умножения значений P и S, можем определить категорию риска. Категории рисков подразделяются на следующие: низкие (R< 6); умеренные ($6 \leq R \leq 12$); высокие (R > 12). Риски в категории «низкие» - допустимы и управляемы в соответствии с существующими в организации мерами (имеются в наличии необходимые процедуры и инструкции, оборудование поддерживается в технически исправном состоянии, своевременно проводится обучение, инструктаж и проверка знаний работников). Риски в категории «умеренные» и «высокие» считают недопустимыми и требуют разработки мер по управлению ими.

Карта опасностей и рисков представлена ниже (таблица 6.4).

Таблица 6.4 - Карта управления (умеренными) рисками

Профессия, должность	Вид деятельности	Идентификационная опасность	Серьёзность последствий возникновения опасности, S	Вероятность возникновения опасности, P.	Риск, R	Осуществляемые меры управления	Рекомендуемые действия	Срок исполнения
1	2	3	4	5	6	7	8	9
Администратор веб-	трудовая	Нервно-психические перегрузки	2	4	8	Инструкция по охране труда при работе с персональными электронно-вычислительными машинами	Самоконтроль	постоянно

Окончание таблицы 6.4

1	2	3	4	5	6	7	8	9
Администратор веб-сервиса	трудовая	Умственное напряжение	2	5	10	Соблюдение распорядка дня	Самоконтроль	постоянно
		Поражение электрическим током	2	3	6	Инструкция по охране труда	Соблюдение и выполнение требований инструкции	постоянно
		Пожарная опасность	1	2	2	Инструкция по пожарной безопасности	Соблюдение правил пожарной безопасности	постоянно
		Напряжение зрительных анализаторов	2	4	8	Инструкция по охране труда при работе с персональными компьютерами	Соблюдение требований инструкции	постоянно
		Статическая поза (заболевания кистей рук)	3	3	9	Самоконтроль	Соблюдение распорядка дня, производственная гимнастика	постоянно
		Простудные заболевания	3	4	12	Самоконтроль	Обеспечение соответствующих условий производственной среды	постоянно

Оценка организации охраны труда, производственной санитарии и промышленной безопасности приведена ниже (таблица 6.5).

Таблица 6.5 - Характеристика производственной санитарии и промышленной безопасности

Исходные параметры	Характеристика реализуемого параметра
1	2
Организационные мероприятия по обеспечению охраны труда	-
Количество имевших место за отчётный период:	-
- аварий/количество пострадавших	-
- инцидентов/количество пострадавших	-
- несчастных случаев/количество пострадавших	-
Технические средства и оборудование, обеспечивающие параметры микроклимата:	
- предусматриваемые системы вентиляции	Естественная
- система отопления в помещении	центральное водяное
- способ уборки помещения	влажная
Технические средства и оборудование, обеспечивающие параметры освещения:	
- характеристика зрительной работы, разряд и подразряд зрительной работы	III
- вид и система искусственного освещения в помещении	общая
- источники искусственного освещения / мощность ламп	9 Вт
- исполнение светильников / количество	светодиодные лампы / 2 шт
- исполнение естественного освещения (боковое или боковое и верхнее)	Боковое
- коэффициент естественной освещенности (КЕО, %)	1,5
- мероприятия по обеспечению нормальной зрительной работы (до нормируемых значений) на рабочих местах	рекомендуется мойка окон 2-4 раза в год
Технические средства и оборудование, обеспечивающие техническую безопасность:	
- знаки безопасности на оборудовании	есть
- класс помещения по опасности поражения электрическим током	без повышенной опасности
- класс электрооборудования по способу защиты человека от поражения электрическим током	I
- сопротивление изоляции токоведущих частей, МОм	0,5

Окончание таблицы 6.5

- тип заземления	T-N
- места (зоны) накопления зарядов статического электричества.	ПЭВМ
- средства технической и коллективной защиты от поражения электрическим током и статического электричества	изоляция, УЗО
- основные и дополнительные электрозащитные средства	-

В соответствии с информацией из приведённой выше таблицы 6.5, представленные мероприятия по обеспечению электробезопасности соответствуют ТКП 181-2009 (02230) «Правила технической эксплуатации электроустановок потребителей» и ТКП 427–2012 «Правила техники безопасности при эксплуатации электроустановок».

Рассчитаем необходимое количество светильников для освещения помещения методом светового потока.

Для расчёта искусственного освещения в цехе методом светового потока используется следующая формула (6.2):

$$N = \frac{E_н \times S \times z \times k}{\eta \times F}, \quad (6.2)$$

где:

N - число светильников, обеспечивающее требуемую освещённость в помещении, шт;

$E_Н$ - нормируемая освещённость (для III разряда зрительной работы и малого, среднего и большого контраста объекта с фоном - 300 лк), лк;

F - световой поток одной лампы (для светодиодной лампы мощностью 9 Вт -700), лм;

S - площадь помещения (25,6 м^2), м^2;

k - коэффициент запаса, зависящий от состояния воздушной среды в помещении (примем равным 1);

z - поправочный коэффициент, учитывающий неравномерность освещённости в помещении (примем равным 1,2);

η - коэффициент использования светового потока, зависит от типа светильника, индекса помещения i, коэффициентов $\rho_п$, $\rho_{ст}$, и $\rho_р$ отражения потока, стен и рабочей поверхности (в формулу значение коэффициента подставляют в долях единицы).

Индекс помещения определяется по формуле (6.3):

$$i = \frac{a \times b}{h_p \times (a + b)}, \quad (6.3)$$

где:

a и b - длина и ширина помещения (для рассматриваемого помещения - 4,3 и 5,95 м), м;

$h_р$ - высота подвеса светильников (2,2 м), м;

Коэффициент отражения побелённых потолков принимается равным $\rho_п$ = 50 %, стен, покрытых на высоту 1,8 м глазурованной плиткой, $\rho_{ст}$ = 50…70 %. Коэффициент отражения стен и потолка ξ зависит от характера отражающей поверхности: учитывая, что в помещении побелённые стены при незанавешенных окнах и светлый потолок – ξ = 50 %;

Подставляя данные в формулу (6.3) получаем:

$$i = \frac{4{,}3 \times 5{,}95}{2{,}2 \times (4{,}3 + 5{,}95)} = \frac{25{,}6}{22{,}55} = 1{,}135$$

При данном индексе площади помещения и коэффициенте отражения стен и потолка ξ (50 %), коэффициент использования светового потока для светодиодных светильников η составляет 24. Подставляя данные в формулу (6.2) получаем необходимое количество светильников:

$$N = \frac{300 \times 25{,}6 \times 1{,}2 \times 1}{24 \times 700} = 0{,}55 шт$$

Таким образом, принимаем количество светильников – 1 шт. В помещении установлено 2 лампы, значит, количество установленных ламп превышает необходимое. Вывод: одну лампу можно убрать, либо установить лампы с меньшей мощностью.

Система пожарной безопасности – это комплекс экономических, социальных, организационных, научно-технических и правовых мер, а также сил и средств, направленных на предупреждение возможных причин пожаров в помещении.

Возможные причины возникновения пожара: неисправность электропроводки, неосторожное обращение с огнём, нахождение в помещении горюче-смазочных материалов и других легко воспламеняющихся веществ.

В таблице 6.6 отражены основные характеристики организации по степени подверженности пожарам.

Таблица 6.6 - Противопожарные мероприятия

Исходные параметры	Значение реализуемого параметра
Наименование помещения	Кабинет
Категория производства по пожароопасности	Д
Классификация производственного помещения по взрыво- и пожароопасности	–
Характеристика материалов стен по сгораемости	Несгораемая
Степень огнестойкости стен	II R 90-КО
Степень огнестойкости перекрытий	II R 60-КО
Расстояние от наиболее удалённого рабочего места до эвакуационного выхода, м	25
Количество эвакуационных выходов, шт.	1
Автоматические установки огнетушения	–
Тип извещателей о пожаре	дымовой
Первичные средства огнетушения	–

Мероприятия по обеспечению пожарной безопасности соответствуют требованиям ППБ 01-2014, ТНПА противопрожарного нормирования и стандартизации.

Во исполнение Закона Республики Беларусь «О пенсионном обеспечении» все объекты хозяйственной деятельности независимо от формы собственности обязаны проводить не реже одного раза в пять лет аттестацию рабочих мест по условиям труда.

Аттестация проводится в соответствии с Положением о порядке проведения аттестации рабочих мест по условиям труда, утверждённым Постановлением Совета Министров Республики Беларусь от 22.02.2008 г. № 253 и Инструкцией по оценке условий труда при аттестации рабочих мест по условиям труда и предоставлению компенсаций по её результатам, утверждённой Постановлением МТ и СЗ 22.02.2008 г. № 35.

Аттестация рабочих мест по условиям труда - система учёта, анализа и комплексной оценки на рабочих местах всех факторов производственной среды, тяжести и напряжённости трудового процесса, воздействующих на работоспособность и здоровье работника в процессе трудовой деятельности.

В Республике Беларусь условия труда подразделяются на четыре класса: оптимальные, допустимые - относятся к безопасным, вредные и опасные.

Компенсация профессиональных вредностей, а также средства защиты и личная гигиена рабочих представлены в таблице 6.7.

Таблица 6.7 - Компенсация профессиональных вредностей. Средства индивидуальной защиты и личная гигиена работающих

Исходные параметры	Значение реализуемого параметра
1	2
Профессия (должность)	Администратор веб-сервиса
Условия труда	2 класс – допустимые
Продолжительность дополнительного отпуска, дни Пенсионный возраст, лет (2018)	1 (по контракту)

Окончание таблицы 6.7

1	2
— женщин	56
— мужчин	61
Обеспечение ЛПП	
Спецодеждой	–
Спецобувью	–
Средствами индивидуальной защиты органов зрения и дыхания	–
Средства обеззараживания кожи	вода, мыло
Метод обеззараживания кожи	мытье рук
Периодичность медосмотра	1 р. в 2 года

В ходе выполнения данного раздела дипломного проекта была проделана следующая работа:

- дана характеристика объекта с точки зрения охраны труда: условия труда администратора веб-сервиса относятся к допустимым условиям (2 класс), которые характеризуются такими уровнями факторов среды и трудового процесса, которые не превышают установленных гигиенических нормативов для рабочих мест, а возможные изменения функционального состояния организма, возникающие под их воздействием, восстанавливаются во время регламентированного отдыха или к началу следующей смены и не оказывают неблагоприятного действия на состояние здоровья работников в ближайшем и отдаленном периоде;

- разработана карта рисков для администратора веб-сервиса;

- совершена оценка организации охраны труда, производственной санитарии, промышленной и пожарной безопасности.

7 ПРОМЫШЛЕННАЯ ЭКОЛОГИЯ

В данном разделе будут рассмотрены некоторые аспекты промышленной экологии.

Промышленная экология – прикладная наука о взаимодействии промышленности и окружающей среды, и наоборот – влияние условий природной среды на функционирование предприятий и их комплексов.

Общая характеристика экологической деятельности организации приведена в таблице 7.1.

Таблица 7.1 - Общая характеристика экологической деятельности организации

Исходные параметры	Значение реализуемого параметра
Нормативы допустимых выбросов (НДВ) (из экологического паспорта)	не требуется
Объем сброса сточных вод, м3 (из экологического паспорта)	0,14м^3/день
Количество (объем) образования твердых бытовых отходов, т (м3) /день	0,08м^3/ день
Наличие систем очистки воды и сточных вод	Отсутствует
Обращение (утилизация, рециклинг, переработка, захоронение и т. п.) с отходами	Раздельный сбор, складирование в контейнер и вывоз, сдача макулатуры, ежегодно
Мероприятия по энергосбережению	Рациональное использование электроэнергии

Таблица 7.2 содержит экологические аспекты деятельности и виды

воздействия экологических аспектов на окружающую среду, а также возможные мероприятия по сокращению воздействия.

Таблица 7.2 - Экологические аспекты деятельности и виды воздействия экологических аспектов на окружающую среду

Экологический аспект	Воздействие на окружающую среду (работающих)	Предложения по сокращению воздействия
Освещённость	Расход энергии	Рациональное использование электроэнергии
Отработанные лампы	Загрязнение тяжёлыми металлами	Сортировка, централизованный сбор и утилизация
Энергия	Загрязнение атмосферы	Рационально использование, мероприятия по энергосбережению
ЭМП	воздействие ЭМП на работающих	Соблюдение режима труда, современное оборудование
Информация	перенапряжение анализаторов	Более эффективные системы поиска информации
Мусор	Твёрдые отходы производства	Раздельный сбор. Переработка вторичного сырья
Сточная вода (бытовая)	Загрязнение гидросферы	Установка счётчика, фильтра, использование рециркуляции бытовой воды

Схема материальных потоков при работе с веб-сервисом представлена ниже (рисунок 7.1).

Компьютерное оборудование имеет свой срок годности и должно быть утилизировано по его истечению.

Компьютерная техника состоит из разных деталей, которые могут негативно сказаться на экологии территории, поэтому бездумное выбрасывание их на свалку может привести к серьёзным последствиям. Запчасти, в которых есть свинец, ртуть, олово, отравляют почву и атмосферу,

что приводит к гибели живых организмов.

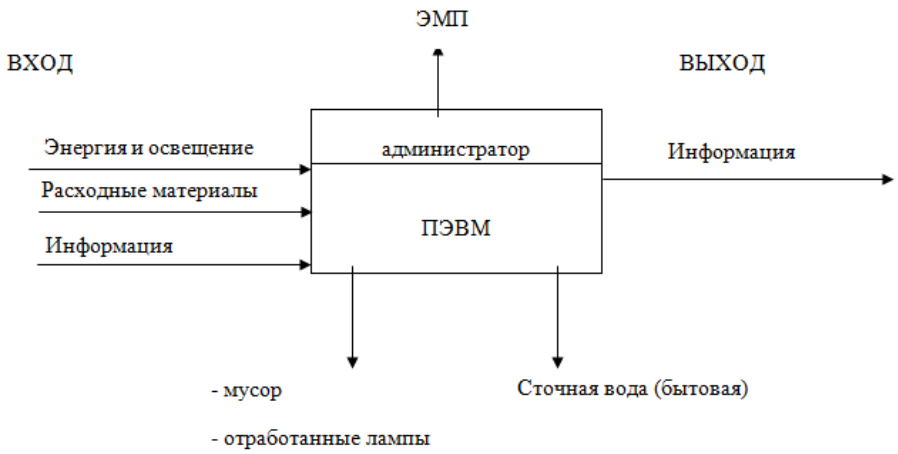


Рисунок 7.1 - Схема материальных потоков при работе с веб-сервисом

В соответствии с Инструкцией о порядке сдачи и приёма лома и отходов, содержащих драгоценные металлы, утверждённой постановлением Министерства финансов Республики Беларусь 31.05.2004 № 87: «Юридические лица независимо от форм собственности и индивидуальные предприниматели обязаны обеспечить полный сбор, первичную обработку, учёт и сдачу всех видов лома и отходов, содержащих драгоценные металлы, во всех местах их хранения и использования, от всех источников образования». В соответствии с пп. 1.1. ст. 17 Закона РБ «Об обращении с отходами» от 20 июля 2007 г. № 271-З «Юридические лица и индивидуальные предприниматели, осуществляющие обращение с отходами, обязаны обеспечивать сбор отходов и их разделение по видам.».

Утилизировать компьютерную технику нужно согласно рекомендации производителя продукта. Компьютеры перерабатываются по определённой схеме: составление паспорта отхода – проведение экологического исследования – разбор техники – сортировка комплектующих – дальнейшая переработка.

На данный момент существует несколько способов утилизации старых компьютеров:

- найти специализированные фирмы по утилизации, которые вывезут и разберут ПК. При этом они должны иметь соответствующую лицензию, паспорт отходов; обязаны заключать договоры о работе и иметь квалифицированный персонал и технику;

- сдать в пункты приёма металла;

- отдать компьютер на запчасти. Стоимость каждого сданного товара может варьироваться в зависимости от сложности его конструкции, количества работающих деталей и расценок, принятых в данной фирме.

В ходе рассмотрения вопросов данного раздела был определён экологический аспект деятельности и виды воздействия экологических аспектов на окружающую среду.

8 РЕСУРСОСБЕРЕЖЕНИЕ

Ресурсосбережение - это процесс рационализации использования материально-технических, трудовых, финансовых, природных и других ресурсов преимущественно на базе интенсификации производства с целью получения продукции с наилучшими качественными показателями и минимумом затрат.

Ресурсосбережение состоит из мероприятий технических, технологических, организационных и экономических. Технические и технологические направления ресурсосбережения отработаны значительно лучше, чем экономические, и выражаются в многочисленных конкретных мероприятиях. Под экономическим механизмом ресурсосбережения понимается система взаимосвязанных экономических элементов, направленных на анализ использования и стимулирование экономии материально-технических, в том числе и топливно-энергетических, трудовых и финансовых ресурсов, внедрение ресурсосберегающих мероприятий, а также обеспечение производства сельскохозяйственной продукции с минимальными затратами всех ресурсов в денежном и натуральном исчислении.

Основная задача ресурсосбережения – экономия материальных ресурсов. Экономия производится различными способами: минимизация использования материальных ресурсов (установка норм) или процесс сбережения их с помощью внедрения новых технологии.

Основные принципы государственной политики Республики Беларусь в сфере ресурсосбережения:

- осуществление государственного надзора за сбережением топливно-энергетических ресурсов;

- разработка государственных и межгосударственных промышленных, республиканских и региональных программ энергосбережения и их финансирование;

- разработка системы финансово-экономических механизмов, которые обеспечивают экономическую заинтересованность производителей и пользователей в эффективном использовании топливно-энергетических ресурсов, вовлечении в топливно-энергетический баланс нетрадиционных и возобновляемых источников энергии, а также в инвестировании средств в энергосберегающие мероприятия;

- повышение уровня самообеспечения республики местными топливно-энергетическими ресурсами;

- осуществление экспертизы энергетической эффективности проектных решений государством;

- создание и распространение экологически чистых, а также безопасных энергетических технологий, обеспечение безопасного для населения состояния окружающие среды в процессе использования топливно-энергетических ресурсов;

- реализация демонстрационных проектов высокой энергетической эффективности;

- информационное обеспечение деятельности по энергосбережению и пропаганда передового отечественного и зарубежного опыта в этой области;

- обучение производственного персонала и населения методам экономии топлива и энергии;

- создание других экономических, информационных, организационных условий для реализации принципов энергосбережения [25].

Кроме того, вопросы энерго- и ресурсосбережения решаются путём принятия и утверждения в республике законов, директив и правовых актов в области рационального использования энергоресурсов и энергосбережения.

Техническими нормативами и правовыми актами по ресурсосбережению

являются государственные, межгосударственные и международные стандарты, строительные нормы и правила, санитарные правила и нормы, гигиенические нормативы, технические кодексы установившейся практики, руководящие документы, правила, инструкции и др.

Во время своей работы компьютерная техника потребляет достаточно большие объёмы электроэнергии, поэтому вопросы реализации ресурсосбережения так важны при работе с компьютером. Основополагающим аспектом в данном случае становится экономия электроэнергии. Большая часть потребляемой компьютером энергии уходит на поддержание функционирования монитора и жёсткого диска. Простейшим способом экономии электроэнергии является отключение персонального компьютера после окончания работы с ним, однако, зачастую случается что компьютер должен оставаться в рабочем состоянии всю ночь или даже несколько суток подряд. Для таких случаев операционные системы реализуют несколько режимов работы.

Для примера рассмотрим режимы работы системы в наиболее популярной операционной системе «Windows». Начиная с версии «Windows 7» операционная система содержит тир энергосберегающих режима - сон, гибернация(спящий режим) и гибридный сон:

- спящий режим максимально снижает электропотребление системы, перемещая все ваши открытые приложения и документы в оперативную память компьютера. На первый взгляд, в этом режиме компьютер выглядит как отключённый, определить факт его работы можно лишь по горящему индикатору питания и продолжающим охлаждать корпус вентиляторам. Благодаря использованию оперативной памяти пользователь может практически моментально включить компьютер и продолжить работу с того места, на котором он остановился. Единственным минусом является то, что можно потерять несохраненные данные открытых приложений при мгновенном обесточивании системы из-за того, что вся информация в этот

момент находится в оперативной памяти;

- режим гибернации по своим функциям аналогичен спящему режиму с то лишь разницей, что все данные заносятся не в оперативную память, а на жёсткий диск, который не является энергозависимым устройством. Таким образом, гибернация устраняет недостаток спящего режима, ведь вы не потеряете несохраненные данные при обесточивании системы. Но из-за этого переход в режим гибернации и выход из него происходит медленнее чем при использовании спящего режима;

- режим гибридного сна объединяет в себе принципы сохранения информации обоих предыдущих режимов и при выходе из гибридного сна система пытается восстановить данные наиболее быстрым способом, то есть, из оперативной памяти, но если информация была повреждена по какой-либо причине, то информация восстанавливается с жёсткого диска. Таким образом пользователь в любом случае не потеряет свою информацию и сохраняется скорость перехода между режимами.

Независимо от версии «Windows» переход в один из вышеописанных режимов выполняется одинаково, через переход в меню «Пуск».

Также в «Windows» кроме ручной активации сберегающих режимов предусмотрен ещё один достаточно мощный инструмент по уменьшению энергопотребления - смена планов электропитания.

Настройка плана электропитания производится в панели управления компьютером, в разделе электропитание. Пользователю предоставлен выбор из 2 основных и 1 дополнительного планов электропитания:

- экономичный режим сохраняет энергию за счёт производительности системы и яркости экрана. Этот план позволяет потреблять меньше электроэнергии;

- производительный режим максимизирует яркость экрана и, при определённых обстоятельствах, может увеличивать производительность и быстродействие компьютера. Использование этого плана потребляет гораздо

больше электроэнергии из сети;

- сбалансированный вариант предлагает полную производительность, когда это необходимо, и сохраняет энергию в период бездействия системы. Этот план подходит большинству пользователей «Windows» [26].

Таким образом, любой пользователь может выбрать и настроить план энергопотребления компьютера под себя, а также, ответственно используя персональный компьютер и его энергосберегающие режимы можно значительно уменьшить энергопотребление системы.

ЗАКЛЮЧЕНИЕ

Результатом выполнения дипломного проектирования является разработанный веб-сервис для записи на приём к врачу и управления этими посещениями. Следующая работа была проведена для достижения поставленных целей:

- проведён анализ предметной области приложения;
- разработана система требований к проектируемой системе;
- описаны аналоги разрабатываемого сервиса;
- проведено проектирование структур хранения данных в приложении;
- разработана архитектура программной системы, а также её компонентов;
- спроектирован и разработан интерфейс программы;
- выбраны инструментальные средства для разработки программной системы;
- описаны варианты использования разрабатываемой программы;
- проведено модульное, функциональное и прочие тестирования системы;
- исследована экономическая эффективность разработанной программы;
- рассмотрены вопросы охраны труда, промышленной экологии и ресурсосбережения.

В первом разделе был осуществлён анализ объекта: проведено описание предметной области и построение её концептуальной модели.

Во втором разделе были разработаны требования к системе: определён список требований и описаны аналоги программной системы.

В третьем разделе было проведено проектирование: структур хранения данных, разработаны архитектуры программной системы и её компонентов, а также разработан пользовательский интерфейс программы.

В четвёртом разделе была проведена реализация программы: выбраны инструментальные средства для разработки, описаны варианты использования и проведены модульное, функциональное и прочие виды тестирования.

В пятом разделе были исследованы экономические аспекты разработанной системы: проведён расчёт трудоёмкости разработки, сметы затрат на разработку, накладных расходов, экономического эффекта от разработки для разработчика ПО.

В шестом разделе были исследованы методы для обеспечения безопасной работы, проведена разработка характеристики объекта с точки зрения охраны труда, разработана карта рисков для администратора веб-сервиса, проведена оценка охраны труда, производственной санитарии, промышленной и пожарной безопасности.

В седьмом разделе определены экологические аспекты деятельности и виды воздействия экологических факторов на окружающую среду.

В восьмом разделе описаны требования к ресурсосбережению.

В результате разработки дипломного проекта были выполнены все поставленные требования к системе.

Таким образом, содержание дипломного проекта показывает, что поставленные задачи к проекту были полностью решены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Microsoft SQL Server. Краткое описание [Электронный ресурс] / Режим доступа: https://flexberry.github.io/ru/gbt_mssql.html - Дата доступа 02.04.2022

2. SQL Server Management Studio — единое средство управления и среда разработки в MS SQL Server 2012 [Электронный ресурс] / Режим доступа: https://tavalik.ru/sql-server-management-studio/ - Дата доступа: 03.04.2022

3. Redux [Электронный ресурс] / Режим доступа: https://blog.skillfactory.ru/glossary/redux/ - Дата доступа: 05.04.2022

4. Что такое n-уровневая архитектура? [Электронный ресурс] / Режим доступа: https://ru.theastrologypage.com/n-tier-architecture - Дата доступа: 06.04.2022

5. 4 типа архитектуры программного обеспечения [Электронный ресурс] / Режим доступа: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724 - Дата доступа: 07.04.2022

6. Что такое Bootstrap [Электронный ресурс] / Режим доступа: http://web.spt42.ru/index.php/chto-takoe-bootstrap - Дата доступа 09.04.2022

7. Краткий обзор языка C# [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ - Дата доступа: 11.04.2022

8. Язык программирования C#: краткая история, возможности и перспективы [Электронный ресурс] / Режим доступа: https://timeweb.com/ru/community/articles/chto-takoe-csharp - Дата доступа: 14.04.2022

9. C# - Преимущества и недостатки [Электронный ресурс] / Режим доступа:

https://shwanoff.ru/plus-minus-c-sharp/ - Дата доступа: 15.04.2022

10. ASP.NET Core - Обзор [Электронный ресурс] / Режим доступа: https://coderlessons.com/tutorials/microsoft-technologies/izuchite-asp-net-core/asp-net-core-obzor/ - Дата доступа: 17.04.2022

11. Обзор ASP.NET [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/overview - Дата доступа: 18.04.2022

12. Описание среды разработки MS Visual Studio [Электронный ресурс] / Режим доступа: https://prog.bobrodobro.ru/63233 - Дата доступа: 21.04.2022

13. Что такое React и почему он так популярен [Электронный ресурс] / Режим доступа: https://liquidhub.ru/blogs/blog/chto-takoe-react - Дата доступа: 23.04.2022

14. Знакомство с ReactJS на базовом уровне [Электронный ресурс] / Режим доступа: https://nuancesprog.ru/p/13297/ - Дата доступа: 24.04.2022

15. TypeScript [Электронный ресурс] / Режим доступа: https://blog.skillfactory.ru/glossary/typescript/ - Дата доступа: 25.04.2022

16. Что такое код Visual Studio? [Электронный ресурс] / Режим доступа: https://ru.education-wiki.com/3958588-what-is-visual-studio-code - Дата доступа: 27.04.2022

17. use case (вариант использования) [Электронный ресурс] / Режим доступа: https://openu.ru/Books/UML/Use_case.asp - Дата доступа: 28.04.2022

18. Реализация прикладного уровня для микрослужб с помощью веб-API [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/microservice-application-layer-implementation-web-api - Дата доступа: 29.04.2022

19. Модульное тестирование с помощью xUnit в ASP.NET Core MVC [Электронный ресурс] / Режим доступа: https://forproger.ru/article/modulnoe-testirovanie-s-pomoshhyu-xunit-v-aspnet-core-mvc - Дата доступа: 02.05.2022

20. Примеры использования Moq [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/150859/ - Дата доступа: 04.05.2022

21. Cypress [Электронный ресурс] / Режим доступа: https://bizzapps.ru/p/cypress/ - Дата доступа: 07.05.2022

22. Тестирование безопасности [Электронный ресурс] / Режим доступа: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/testirovanie-bezopasnosti-2 - Дата доступа: 08.05.2022

23. Аутентификация с помощью ASP.NET Identity [Электронный ресурс] / Режим доступа: https://professorweb.ru/my/ASP_NET/identity/level1/1_5.php - Дата доступа: 09.05.2022

24. Трудовой кодекс Республики Беларусь с обзором изменений, внесенных Законом Республики Беларусь от 20 июня 2007 г. № 272-З-Минск: Амалфея, 2007. - 288 с.

25. Основные положения нормативно-правовой базы энергопотребления и энергосбережения [Электронный ресурс] / Режим доступа: http://www.technopark.by/files/gl10.htm - Дата доступа: 12.05.2022

26. Способы снижения расходов компьютера на электроэнергию [Электронный ресурс] / Режим доступа: https://php64.ru/poleznoe/ekonomim-elektrichestvo-kompyutera.php - Дата доступа: 12.05.2022

Листинг А.1 - Метод «быстрого» поиска докторов

```
      public async Task<IEnumerable<Doctor>>
GetFastSearchedDoctorsAsync(string            searchTerm)
      {
            var values = searchTerm.Trim().Split(" ");

            var doctorComparer = new DoctorComparer();

                  var doctors = new HashSet<Doctor>(doctorComparer);

            foreach(var value in values)
                  {
                        IEnumerable<Doctor> temp = await FindByCondition(doc
=>
      doc.User.FirstName.ToLower().Contains(value.ToLower())
                        ||
doc.User.LastName.ToLower().Contains(value.ToLower())
                  || doc.User.Country.ToLower().Contains(value.ToLower())
                        ||
doc.User.Country.ToLower().Contains(value.ToLower())
                  || doc.Specialization.Name.ToLower()
                  .Contains(value.ToLower())).ToListAsync();

                        foreach (var item in temp) doctors.Add(item);
            }

            return doctors;
      }
```

Листинг А.2 - Метод регистрации нового пользователя

```
      public async Task<AuthenticationResultDto>
            RegisterAsync(UserRegistrationDto registrationDto)
      {
            var existingUser = await
                  _userManager.FindByEmailAsync(registrationDto.Email);

            if (existingUser != null) return new AuthenticationResultDto
            {
                  ErrorMessages = new List<string> { "Данная электронная
почта                        уже зарегистрирована, попробуйте другую!" }
            };

            existingUser = await _userManager.
                  FindByNameAsync(registrationDto.UserName);

            if (existingUser != null) return new AuthenticationResultDto
            {
```

```
                ErrorMessages = new List<string> { "Данное имя пользователя
                    уже зарегистрировано!" }
            };

            var newUser = _mapper.Map<User>(registrationDto);
            if (registrationDto.Password != registrationDto.ConfirmPassword)
                return new AuthenticationResultDto
                {
                    ErrorMessages = new List<string> { "Пароль
подтверждён                         неверно!" }
                };

            var createdUser = await _userManager.CreateAsync(newUser,
                registrationDto.Password);

            if (!createdUser.Succeeded)
                return new AuthenticationResultDto
                {
                    ErrorMessages = createdUser.Errors.Select(err =>
                        err.Description)
                };

            if (!string.IsNullOrEmpty(newUser.Type))
            {
                var result = await _userManager.AddToRoleAsync(newUser,
                    newUser.Type);
                if(!result.Succeeded)
                return new AuthenticationResultDto
                {
                    ErrorMessages = result.Errors.Select(err =>
                        err.Description)
                };
            }

            var token = await
                _userManager.GenerateEmailConfirmationTokenAsync(newUser);
            var confirmationLink = $"http://localhost:3000/emailConfirmed?
                token={token}&email={newUser.Email}";

            var message = new MessageOptions(new string[] { newUser.Email },
                "Письмо подтверждения", confirmationLink);

            await _emailService.SendConfirmEmail(message);

            return await GenerateAuthResultForUser(newUser);
        }
```

Листинг А.3 - Метод вычисления списка доступного времени для записи к доктору на определённую дату

```
        public async Task<IEnumerable<AvailableAppointmentTimeDto>>
            Handle(GetAvailableAppointmentTimeForDateCommand command,
            CancellationToken cancellationToken)
```

```
      {
            var availableTime = new List<AvailableAppointmentTimeDto>();

            var doctor = await

      _repository.Doctor.GetDoctorByDoctorIdAsync(command.doctorId);

            if (doctor == null) return availableTime;

            var doctorAppointments = await _repository.Appointment.
                  GetAllAppointmentsForDoctorAsync(command.doctorId);

            var appointmentsTime = doctorAppointments.Where(app =>
                  app.StartDate.Date.Equals(command.date.Date)).
                  Select(app => app.StartTime);

            var doctorScedule = await _repository.DoctorsScedule.
                  GetDoctorsScedule(command.doctorId);

            var doctorWorkingDays = string.Join(" ", doctorScedule.
                  Select(sced => sced.dayOfWeek.ToString()));

           if  (!doctorWorkingDays.Contains(command.date.DayOfWeek.ToString())
                  || command.date.Date < DateTime.UtcNow.Date)
                  return availableTime;

            var doctorDayOffs = await _repository.DoctorDayOff.
                  GetDoctorsDayOffsByDoctorId(command.doctorId);

            var unavailableDays = doctorDayOffs.Where(day => day.Status !=
                  DoctorStatusEnum.Available).Select(day => day.Date);

            foreach (var date in unavailableDays)
                  if (command.date.Date.Equals(date.Date)) return
availableTime;

            var time = doctor.WorkDayStart;

            while(time < doctor.WorkDayEnd)
            {
                  if (DateTime.UtcNow.Date == command.date.Date &&
                        DateTime.Now.TimeOfDay > time)
                  {
                        time = time.Add(new TimeSpan(0,
                              doctor.AverageAppointmentTime, 0));
                        continue;
                  }

                  bool available = true;

                  foreach (var appTime in appointmentsTime)
                        if (time >= appTime && time <= appTime.Add(new
                        TimeSpan(0, doctor.AverageAppointmentTime, 0)))
                                    available = false;
```

```
                if (available && time > doctor.DinnerStart.Subtract(new
                TimeSpan(0, doctor.AverageAppointmentTime, 0)) && time <
                doctor.DinnerEnd) available = false;

                if (available) availableTime.Add(new
                AvailableAppointmentTimeDto
                {
                      doctorId = command.doctorId,
                      Time = $"{time.Hours}:{time.Minutes}"
                });

                time = time.Add(new TimeSpan(0,
                      doctor.AverageAppointmentTime, 0));
          }

          return availableTime;
    }
```

Листинг А.4 - Тест GetDoctorsCommand_ShouldReturnAllDoctors

```
    [Fact]
    public async Task GetDoctorsCommand_ShouldReturnAllDoctors()
    {
          var command = new GetDoctorsCommand()
          {
                doctorParameters = new DoctorParameters { }
          };

          var doctorMock = new Mock<IDoctorRepository>();
          doctorMock.Setup(repo =>
                repo.GetAllDoctorsAsync(command.doctorParameters,
                false)).Returns(TestDoctors());

          var mockRepoManager = new Mock<IRepositoryManager>();
          mockRepoManager.Setup(repo =>
                repo.Doctor).Returns(doctorMock.Object);

          var mockConfigurationRoot = new Mock<IConfigurationRoot>();
          mockConfigurationRoot.SetupGet(config =>
                config[It.IsAny<string>()]).Returns("some setting");

          var mapperMock = new Mock<IMapper>();
          mapperMock.Setup(mapper => mapper.Map<IEnumerable<DoctorDto>>(It.
                IsAny<IEnumerable<Doctor>>())).Returns(TestDoctorsDto());

          var commandHandler = new
                GetDoctorsCommandHandler(mockRepoManager.Object,
                mapperMock.Object);

          var doctorsDto = await commandHandler.Handle(command,
                CancellationToken.None);

          Assert.NotNull(doctorsDto);
          Assert.IsAssignableFrom<IEnumerable<DoctorDto>>(doctorsDto);
```

```
        Assert.Equal(TestDoctorsDto().Count(), doctorsDto.Count());
    }
```

Листинг А.5 - Тест GetDoctorCommand_ShouldReturnNull

```
    [Fact]
    public async Task GetDoctorCommand_ShouldReturnNull()
    {
        var command = new GetDoctorCommand()
        {
            doctorId = Guid.NewGuid()
        };

        var doctorMock = new Mock<IDoctorRepository>();
        doctorMock.Setup(repo =>
            repo.GetDoctorByDoctorIdAsync(command.doctorId,
            false)).Returns(GetTestDoctor(command.doctorId));

        var mockRepoManager = new Mock<IRepositoryManager>();
        mockRepoManager.Setup(repo =>
            repo.Doctor).Returns(doctorMock.Object);

        var mockConfigurationRoot = new Mock<IConfigurationRoot>();
        mockConfigurationRoot.SetupGet(config =>
            config[It.IsAny<string>()]).Returns("some setting");

        var mapperMock = new Mock<IMapper>();
        mapperMock.Setup(mapper =>
            mapper.Map<DoctorDto>(It.IsAny<Doctor>())).
            Returns(GetTestDoctorDto(command.doctorId));

        var commandHandler = new
            GetDoctorCommandHandler(mockRepoManager.Object,
            mapperMock.Object);

        var doctorDto = await commandHandler.Handle(command,
            CancellationToken.None);

        Assert.Null(doctorDto);
    }
```

Листинг А.6 - Тест CreateDoctorCommand_ShouldCreateNewDoctor

```
    [Fact]
    public async Task CreateDoctorCommand_ShouldCreateNewDoctor()
    {
        var command = new CreateDoctorCommand()
        {
            doctor = new DoctorForCreationDto { }
        };

        var doctorId = Guid.NewGuid();
```

```
        var doctorMock = new Mock<IDoctorRepository>();
        doctorMock.Setup(repo => repo.CreateDoctor(new Doctor { Id =
              doctorId }));

        var mockRepoManager = new Mock<IRepositoryManager>();
        mockRepoManager.Setup(repo =>
              repo.Doctor).Returns(doctorMock.Object);
        mockRepoManager.Setup(repo =>
              repo.SaveAsync()).Returns(Task.CompletedTask);

        var mockConfigurationRoot = new Mock<IConfigurationRoot>();
        mockConfigurationRoot.SetupGet(config =>
              config[It.IsAny<string>()]).Returns("some setting");

        var mapperMock = new Mock<IMapper>();
        mapperMock.Setup(mapper =>
              mapper.Map<Doctor>(It.IsAny<DoctorForCreationDto>())).
              Returns(new Doctor { Id = doctorId });

        var commandHandler = new
              CreateDoctorCommandHandler(mockRepoManager.Object,
              mapperMock.Object);

        var id = await commandHandler.Handle(command,
              CancellationToken.None);

        Assert.True(id == doctorId);
    }
```

Листинг А.7 - Тест When doctor go to notifications page, notifications page should appear

```
  it("When doctor go to notifications page, notifications page should appear",
     () => {
        cy.get("body").then((body) => { if (
              body.find("button[type=submit]")[0] &&
                    body.find("button[type=submit]")[0].innerText ===
                    "Войти" )
              {
                    cy.findByLabelText("Имя пользователя или эл.
                                почта:").type( "TestDoctor", );
                    cy.findByLabelText("Пароль:").type("Password123!");
                                cy.contains("Войти").click(); } });
                    cy.contains("Добро пожаловать!");
                    cy.findByText("Уведомления").click();
                    cy.url().should("include", "notifications");
```

```
            }
    );
```

Листинг А.8 - Тест When trying sign in with blank inputs, error messsage should appear

```
it("When trying sign in with blank inputs, error messsage should appear", ()
    => {
          cy.contains("Вход в аккаунт");
          cy.contains("Войти").click();
          cy.contains("Логин обязателен!");
          cy.contains("Пароль обязателен!");
          cy.findByLabelText("Имя пользователя или эл. почта:").type("A");
          cy.findByLabelText("Пароль:").type("A");
          cy.contains("Войти").click();
          cy.contains("Неверные данные пользователя!");
       }
  );
```

Листинг А.9 - Тест When trying register doctor with wrong data in inputs, error messsages should appear

```
it("When trying register doctor with wrong data in inputs, error messsages
    should appear",
() => {
          cy.contains("Вход в аккаунт");
          cy.contains("Зарегистрироваться").click();
          cy.contains("Зарегистрироваться как:");
          cy.contains("Доктор").click();
          cy.get("form").submit();
          cy.contains("Имя обязательно!");
          cy.contains("Фамилия обязательна!");
          cy.contains("Дата рождения обязательна!");
          cy.contains("Дата начала работы обязательна!");
          cy.contains("Почтовый код обязателен!");
          cy.contains("Номер телефона обязателен!");
          cy.contains("Страна обязательна!");
          cy.contains("Город обязателен!");
```

```
        cy.contains("Специализация обязательна!");
        cy.contains("Продолжительность приёма обязательна!");
        cy.contains("Время начала рабочего дня обязательно!");
        cy.contains("Время конца рабочего дня обязательно!");
        cy.contains("Время начала обеденного перерыва обязательно!");
        cy.contains("Время конца обеденного перерыва обязательно!");
        cy.contains("Номер лицензии обязателен!");
        cy.contains("Адрес работы обязателен!");
        cy.contains("Имя пользователя обязательно!");
        cy.contains("Почта обязательна!");
        cy.contains("Пароль обязателен!");
        cy.contains("Подтверждение пароля обязательно!");
        cy.findByLabelText("Имя").type("A", { force: true });
        cy.findByLabelText("Фамилия").type("A", { force: true });
        cy.findByLabelText("Почтовый код").type("A", { force: true });
        cy.findByLabelText("Номер телефона").type("A", { force: true });
        cy.findByLabelText("Страна").type("A", { force: true });
        cy.findByLabelText("Город").type("A", { force: true });
        cy.findByLabelText( "Примерная продолжительность вашего приёма(в
              минутах):", ).type("1", { force: true });
        cy.findByLabelText("Номер лицензии").type("A", { force: true });
        cy.findByLabelText("Адрес работы").type("A", { force: true });
       cy.findByLabelText("Имя пользователя").type("A", { force: true });
        cy.findByLabelText("Эл. почта").type("A", { force: true });
        cy.findByLabelText("Пароль").type("A", { force: true });
        cy.findByLabelText("Подтвердите пароль").type("A", { force:
true });    cy.contains("Имя должно состоять минимум из 2 символов!");
        cy.contains("Фамилия должна состоять минимум из 2 символов!");
        cy.contains("Дата рождения обязательна!"); cy.contains("Дата
начала            работы обязательна!");
        cy.contains("Код должен состоять из 5 цифр!");
        cy.contains("Неверный формат номера!");
        cy.contains("Название страны не может быть меньше 3 символов!");
        cy.contains("Название города не может быть меньше 3 символов!");
        cy.contains("5 минут - минимальное время");
        cy.contains("Имя пользователя должно состоять минимум из 2
              символов!");
```

```
            cy.contains("Неверный формат адреса почты!");
            cy.contains("Пароль должен состоять минимум из 8 символов!");
            cy.findByLabelText("Имя").type( "Аааааааааааааааааааааааааааааа
      аа          аааааааааааааааааааааааааааа", { force: true }, );
            cy.findByLabelText("Фамилия").type( "Аааааааааааааааааааааааааааа
      аа
                  аааааааааааааааааааааааааааа", { force: true }, );
            cy.findByLabelText("Страна").type( "Аааааааааааааааааааааааааааа
      аа          аааааааааааааааааааааааааааа", { force: true }, );
            cy.findByLabelText("Город").type( "Аааааааааааааааааааааааааааа
      аа          аааааааааааааааааааааааааааа", { force: true }, );
            cy.findByLabelText( "Примерная продолжительность вашего приёма(в
                  минутах):", ).type("100", { force: true });
            cy.findByLabelText("Имя

            пользователя").type( "Аааааааааааааааааааааааааааааааааааааа
                  аааааааааааааааааааааааа", { force: true }, );
            cy.findByLabelText("Пароль").type( "Аааааааааааааааааааааааааааа
            аа    аааааааааааааааааааааааааааа", { force: true }, );
            cy.contains("Имя должно состоять максимум из 50 символов!");
            cy.contains("Фамилия должна состоять максимум из 50 символов!");
            cy.contains("Название страны не может быть больше 50 символов!");
            cy.contains("Название города не может быть больше 50 символов!");
            cy.contains("45 минут - максимальное время");
            cy.contains("Имя пользователя должно состоять максимум из 20
                  символов!");
            cy.contains("Пароль должен содержать минимум 1 цифру");
      }
);
```