**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ПП.008 1-40 05 01-01 ПЗ

Разраб.

Провер.

Реценз.

Н. Контр.

Утверд.

Разработка информационной системы записи на прием к врачу

Лит.

Листов

[Введение 3](#_Toc99395208)

[1 Анализ объекта 4](#_Toc99395209)

[1.1 Описание предметной области 4](#_Toc99395210)

[1.2 Построение концептуальной модели предметной области 5](#_Toc99395211)

[2 Постановка задачи 9](#_Toc99395212)

[2.1 Определение требований к программной системе. 9](#_Toc99395213)

[2.2 Описание аналогов системы 11](#_Toc99395214)

[2.3 Обзор и обоснование выбора инструментальных средств 17](#_Toc99395215)

[3 Проектирование 23](#_Toc99395216)

[3.1 Разработка архитектуры программного продукта 23](#_Toc99395217)

[3.2 Проектирование структур хранения данных 28](#_Toc99395218)

[3.3 Описание реализации вариантов использования 30](#_Toc99395219)

[4 Реализация 34](#_Toc99395220)

[4.1 Разработка классов информационной системы 34](#_Toc99395221)

[4.2 Разработка интерфейса программного продукта 43](#_Toc99395222)

[4.3 Разработка алгоритмов реализации вариантов использования 54](#_Toc99395223)

[4.4 Модульное тестирование алгоритмов реализации вариантов использования 56](#_Toc99395224)

[5 Системное тестирование 59](#_Toc99395225)

[5.1 Функциональное тестирование 60](#_Toc99395226)

[5.2 Оценка безопасности 61](#_Toc99395227)

[6 Экономическая часть 65](#_Toc99395228)

[6.1 Расчет трудоемкости разработки программного обеспечения 65](#_Toc99395229)

[6.2 Определение стоимости разработки программного обеспечения 65](#_Toc99395230)

[6.3 Расчёт экономического эффекта от внедрения программного продукта 75](#_Toc99395231)

[7 Охрана труда 78](#_Toc99395232)

[8 Промышленная экология 92](#_Toc99395233)

[9 Ресурсосбережение 96](#_Toc99395234)

[Заключение 100](#_Toc99395235)

[Список использованной литературы 102](#_Toc99395236)

# Введение

В современном мире информационные технологии активно внедряются во все сферы жизни, привнося значительные изменения и улучшения в повседневную деятельность людей. Одной из таких сфер является медицинская отрасль, которая сталкивается с проблемами, связанными с организацией работы, управлением пациентами и предоставлением услуг. Важной задачей в этой области является оптимизация процесса записи на прием к врачу, который зачастую является неэффективным, затрудняет работу медицинского персонала и вызывает неудобства для пациентов.

Целью данной дипломной работы является разработка информационной системы записи на прием к врачу, которая бы обеспечивала удобство и быстроту записи для пациентов, а также упрощала и оптимизировала работу медицинского персонала. Для достижения этой цели предполагается выполнить следующие задачи:

1. Изучить и проанализировать существующие методы и системы записи на прием к врачу.
2. Определить основные требования к функциональности и взаимодействию пользователей с информационной системой.
3. Разработать архитектуру и проектную документацию информационной системы.
4. Выбрать необходимые технологии и инструменты для реализации системы.
5. Реализовать и протестировать информационную систему, а также провести ее апробаци.

Ожидается, что разработанная информационная система упростит и ускорит процесс записи на прием к врачу, повысит уровень удовлетворенности пациентов и оптимизирует работу медицинского персонала.

# 1 Анализ объекта

## 1.1 Описание предметной области

Предметной областью данной дипломной работы является медицинская сфера, а точнее, процесс записи на прием к врачу в медицинских учреждениях. В данной главе рассматривается структура и особенности работы медицинских организаций, а также основные проблемы, возникающие при организации записи на прием к врачу.

Медицинские учреждения – это организации, предоставляющие медицинские услуги населению. К ним относятся поликлиники, больницы, диагностические центры и другие специализированные учреждения. Основными участниками процесса оказания медицинских услуг являются пациенты и медицинский персонал. Медицинский персонал включает в себя врачей различных специальностей, медицинских сестер, администраторов и других сотрудников [1].

Одной из важнейших задач, стоящих перед медицинскими учреждениями, является организация эффективного и удобного процесса записи на прием к врачу. Важно обеспечить равный доступ к медицинским услугам для всех пациентов, а также сократить время ожидания и упростить процесс записи.

Традиционными методами записи на прием к врачу являются:

1. Запись через регистратуру медицинского учреждения. Пациенты обращаются в регистратуру, где администратор осуществляет запись на прием к врачу на определенное время и дату.
2. Запись по телефону. Пациенты звонят в медицинское учреждение и согласовывают время приема с администратором.

Однако, традиционные методы записи на прием к врачу обладают рядом недостатков:

1. Очереди и длительное время ожидания в регистратуре.
2. Затруднения при записи по телефону, связанные с недоступностью линий, занятостью администратора и т.д.
3. Ошибки при записи, связанные с человеческим фак тором, такие как неправильное указание времени приема, даты или специалиста. 4. Отсутствие возможности управления записью на прием в удобное для пациента время и из любого места.
4. Недостаточная информированность пациентов о доступных врачах, их специализации и расписании работы.

В связи с этим, возникает необходимость внедрения современных информационных технологий, которые могут улучшить и оптимизировать процесс записи на прием к врачу. Разработка информационной системы записи на прием к врачу предполагает решение следующих проблем:

1. Упрощение и ускорение процесса записи на прием для пациентов.
2. Уменьшение вероятности ошибок, связанных с человеческим фактором, при записи на прием.
3. Улучшение информированности пациентов о доступных врачах, их специализации, расписании работы и свободных временных слотах для записи.
4. Оптимизация работы медицинского персонала, связанной с организацией записи на прием и ведением учета пациентов.
5. Обеспечение возможности удаленного доступа к информационной системе для пациентов и медицинского персонала.

Таким образом, разработка информационной системы для записи на прием к врачу является актуальной и важной задачей, которая способствует повышению качества предоставления медицинских услуг, удовлетворенности пациентов и эффективности работы медицинского персонала.

## 1.2 Построение концептуальной модели предметной области

Включают широкий спектр символических моделей:

* модель жизненного цикла систем, описывающую процессы существования системы от зарождения замысла до прекращения функционирования [2];
* модели операций, выполняемых объектом и представляющих описание взаимосвязанной совокупности процессов функционирования отдельных элементов объекта при реализации тех или иных функций объекта (в их состав могут входить модели надежности, характеризующие выход элементов системы из строя под влиянием эксплуатационных факторов, и модели живучести, характеризующие выход элементов системы из строя под влиянием целенаправленного воздействия внешней среды);
* информационные модели, отображающие во взаимосвязи источники и потребители информации, виды информации, характер ее преобразования, а также временные и количественные характеристики данных;
* процедурные модели, описывающие порядок взаимодействия элементов исследуемого объекта при выполнении различных операций (например, обработка материалов, деятельность персонала, использование информации, в том числе и различные процедуры принятия управленческих решений);
* временные модели, описывающие процедуру функционирования объекта во времени распределения ресурса «время» по отдельным компонентам объекта.

Модель обычно является комбинацией физических, математических и гибридных элементов либо целиком реализована в виде компьютерной имитации.

На практике для прогнозирования ожидаемого поведения сложной системы в целом обычно нецелесообразно строить модель функционирования, способную отразить все детали поведения системы. Поэтому при испытаниях системы в целом ее наблюдаемое поведение обычно анализируется на двух уровнях. Первый уровень относится к сквозным показателям функционирования, определенным в требованиях к системе. Второй - к тем подсистемам и компонентам, которые характеризуются каким-то критическим поведением. Последний уровень особенно важен, когда сквозное испытание не дает ожидаемого результата и требуется найти источник отклонений.

Решение о том, насколько детальной должна быть модель, применяемая для испытаний на уровне системы, является главным образом функцией системной инженерии, поскольку требуется искать компромисс между риском, сопутствующим игнорированию включения в модель некоторых функций, и ценой их включения.

При построении концептуальной модели предметной области web-приложения была разработана функциональная структура приложения, состоящая из следующих подсистем:

1. Подсистема работы с пользователями. Эта подсистема предназначена для управления учетными записями пользователей, включая регистрацию, аутентификацию и авторизацию, а также редактирование профиля пользователя.
2. Подсистема записи на прием. Она позволяет пользователям просматривать историю своих записей на прием, а также регистрировать новый запрос на прием.
3. Подсистема категорий. Эта подсистема отвечает за работу с категориями врачей, на которые будет доступна запись.
4. Подсистема управления записями на прием для работников регистратуры. Эта подсистема предоставляет возможность просматривать список записей, изменять их статусы для отслеживания процесса обработки и при необходимости изменять данные самой записи.
5. Подсистема администрирования. Она позволяет администраторам просматривать список пользователей и менеджеров, а также добавлять и изменять данные пользователей.

Концептуальная модель предметной области включает в себя разработку и анализ функциональных возможностей каждой из этих подсистем, а также определение взаимосвязей между ними. При этом учитываются требования к системе, выявленные на этапе анализа предметной области и анализа аналогов.

# 2 Постановка задачи

## 2.1 Определение требований к программной системе.

Перед тем как начать разрабатывать программное средство необходимо проанализировать требования к программному средству и поставить задачи.

Цель процесса анализа требований к программному средству заключается в установлении и документировании требований к программному обеспечению.

Разработка программного средства будет вестись на операционной системе Windows 10.

При разработке программного средства будут использованы следующие технологии: язык программирования Java, сервер базы данных H2. В качестве фреймворка будет выбран фреймворк Spring Framework.

Для реализации Back-end части будут использована технология Spring Framework.

Для реализации Front-end части будут использованы следующие технологии: Thymeleaf, HTML, CSS и JavaScript.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Существует огромное количество инструментария для программирования как на стороне сервера, так и на стороне клиента, как платных, так и бесплатных.

Основные программные средства, которые будут использованы при разработке программного средства:

интегрированная среда разработки Eclipse для разработки на языках программирования Java и JavaScript, языке разметки HTML и каскадных таблиц стилей CSS с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами рефакторинга для Java;

браузеры Opera и Google Chrome для проверки на кроссбраузерность программного средства, а также проверки результатов работы программного средства при разработке.

Все перечисленные выше серверные и клиентские технологии, а также соответствующий инструментарий идеально подходят для разработки веб-приложений.

Любой цикл разработки программного средства начинается с анализа требований. Цель этой стадии – определение детальных требований к программному средству [3].

Основные требования к программному средству системы управления персоналом предприятия представлены в таблице 1.

Таблица 1 - Основные требования к программному средству

|  |  |
| --- | --- |
| Вид требования | Содержание требования |
| Бизнес-требование (*B*1) | Все формы программного средства должны быть удобными для пользователей |
| Бизнес-требование (*B*2) | Для эксплуатации интерфейса программного средства от пользователей не должно требоваться специальных технических навыков, знания технологий или программных продуктов, за исключением общих навыков работы с персональным компьютером и стандартным веб-браузером |
| Нефункциональное требование (*NF*1) | Программное средство должно быть адаптивно под любое устройство |
| Нефункциональное требование (*NF*2) | Цветовая гамма программного средства должна иметь положительные цвета |
| Функциональное требование (*F*1) | В программном средстве должно быть предусмотрено разделение ролей на гостя, пользователя и администратора |
| Функциональное требование (*F*2) | Не авторизовавшись в системе, пользователь должен иметь доступ к навигационной панели, которая обеспечивает переход к основным пунктам меню |
| Функциональное требование (*F*3) | Графическая оболочка страниц должна делиться на следующие разделы:  – навигационное меню;  – ссылка на главную страницу;  – поле для отображения контента выбранной страницы |
| Функциональное требование (*F*4) | Каждому пользователю, в зависимости от роли, показывается свое меню |

Требования к программному средству определены

## 2.2 Описание аналогов системы

В данной главе представлены пять аналогов разрабатываемой информационной системы записи на прием к врачу. Для каждого аналога приведены ссылки, описание, а также список преимуществ и недостатков.

**DocDoc**

Ссылка: <https://www.docdoc.ru/>

DocDoc – это популярная российская сервисная платформа, предоставляющая возможность пациентам быстро и удобно записаться на прием к врачу через интернет. Система DocDoc предлагает широкий выбор специалистов различных медицинских учреждений, предоставляя информацию о врачах, их опыте работы, отзывах пациентов и расписании [4].

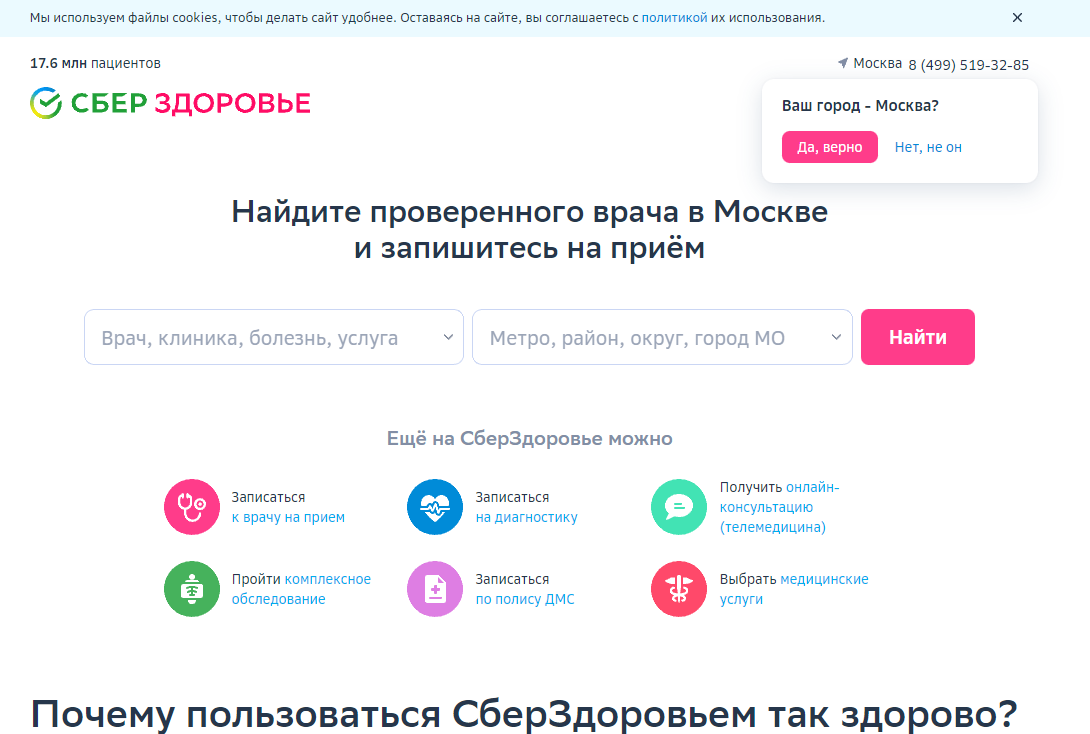


Рисунок 2.1 - DocDoc

Преимущества:

* Большой выбор врачей и медицинских учреждений.
* Удобный поиск по специализации, геолокации и времени приема.
* Отзывы пациентов о врачах, что помогает сделать осознанный выбор.
* Бесплатное использование сервиса для пациентов.

Недостатки:

* Сервис может не включать в себя все медицинские учреждения и врачей, работающих в определенном регионе.
* Возможность записаться только к тем врачам, которые сотрудничают с платформой.

**Zocdoc**

Ссылка: <https://www.zocdoc.com/>

Zocdoc – это американский сервис для записи на прием к врачу, который предлагает удобный интерфейс для поиска и выбора специалиста. Пациенты могут просматривать расписание врачей, выбирать свободное время и осуществлять запись на прием через интернет. Система также предлагает возможность отмены или изменения записи, если пациенту это необходимо [5].

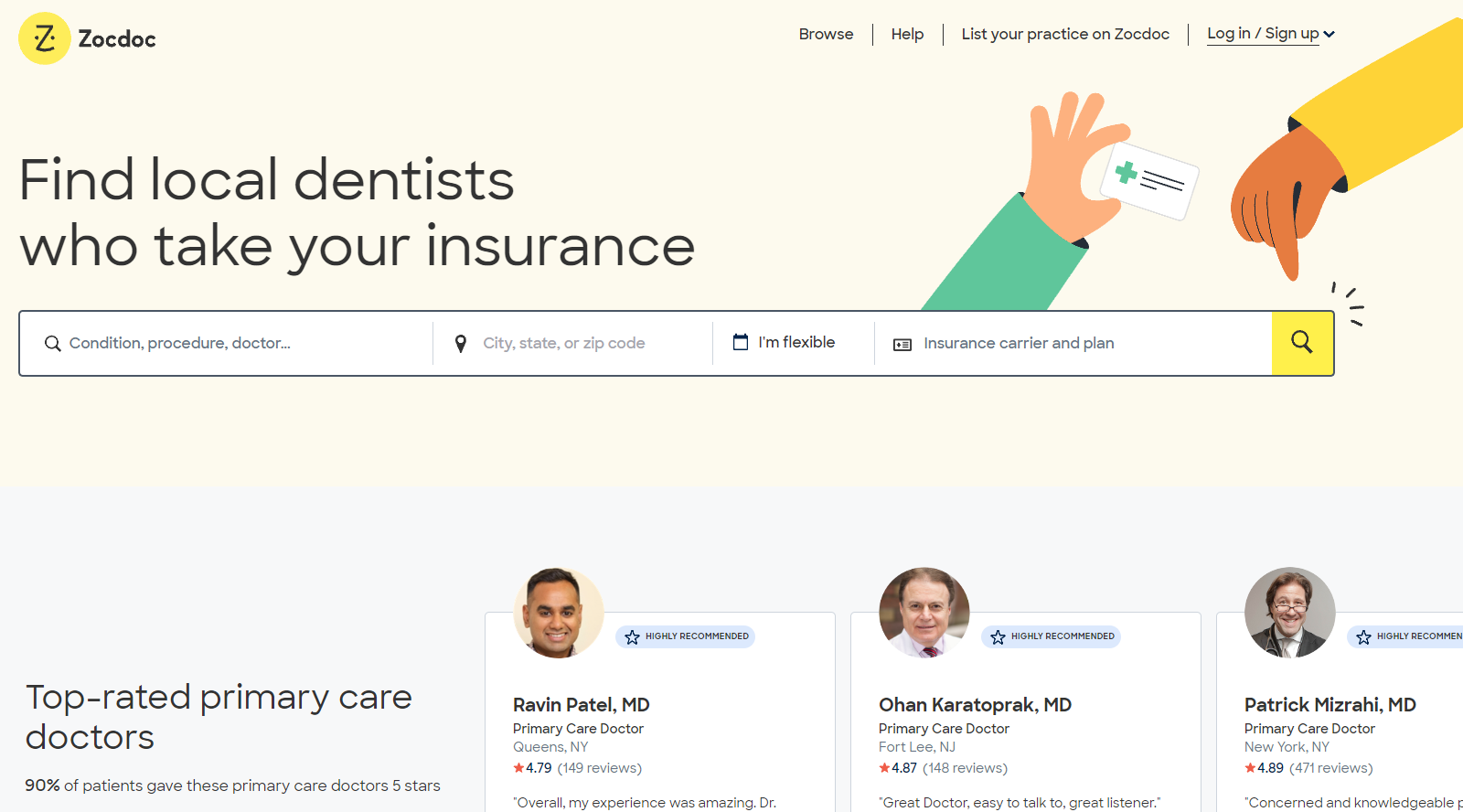


Рисунок 2.2 - Zocdoc

Преимущества:

* Удобный интерфейс для поиска и записи на прием к врачу.
* Возможность отмены или изменения записи.
* Широкий выбор специалистов и медицинских учреждений.
* Наличие мобильного приложения для iOS и Android.

Недостатки:

* Ограниченная география использования сервиса (в основном США).
* Возможность записи только к тем врачам, которые сотрудничают с платформой.

**Medesk**

Ссылка: <https://www.medesk.ru/>

Medesk – это российская медицинская информационная система, которая предоставляет инструменты для управления медицинским учреждением и записи на прием к врачу. Медицинские организации могут использовать эту платформу для автоматизации своих процессов и предоставления возможности пациентам записаться на прием через интернет [6].

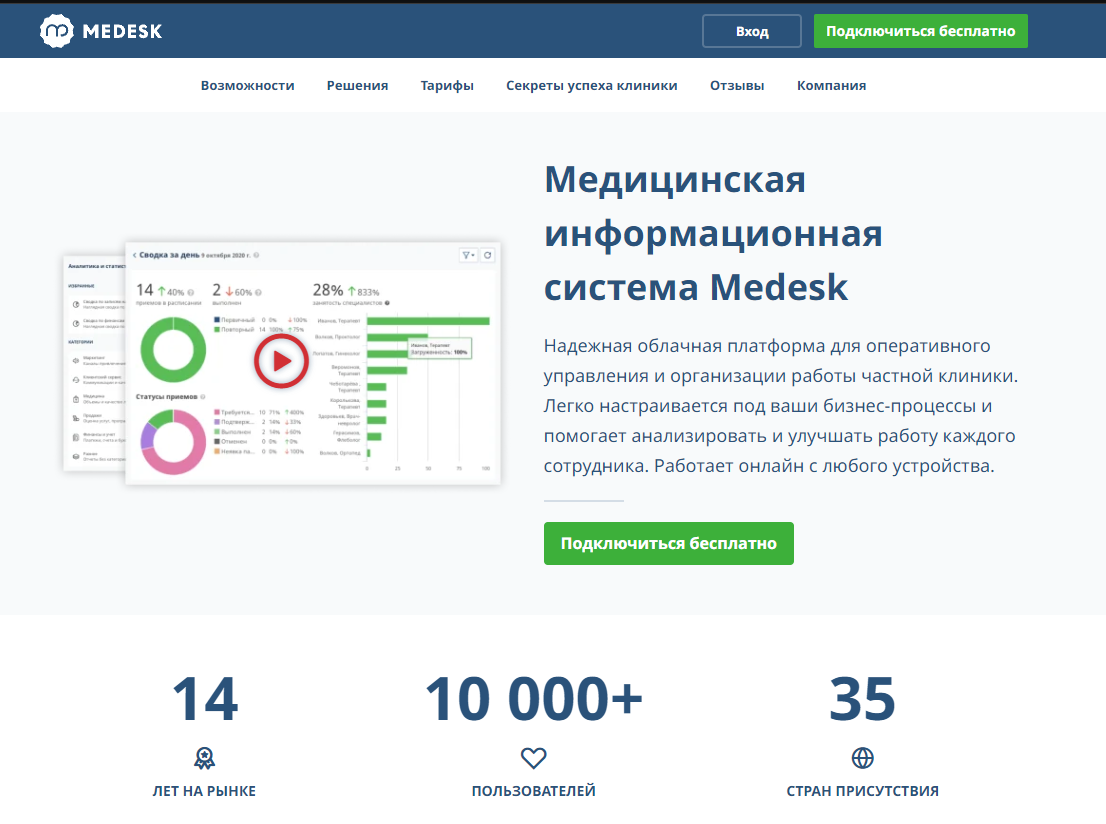


Рисунок 2.3 - Medesk

Преимущества:

* Интеграция с внутренними процессами медицинского учреждения.
* Удобный интерфейс для записи на прием к врачу.
* Возможность интеграции с другими информационными системами.
* Наличие мобильного приложения для iOS и Android.

Недостатки:

* Сложность использования для малых медицинских учреждений, не имеющих специалистов в области IT.
* Необходимость оплаты лицензии для использования системы.

**Bookimed**

Ссылка: <https://bookimed.com/>

Bookimed – это международная платформа, предоставляющая услуги по организации медицинской помощи за рубежом. Сервис помогает пациентам найти подходящего врача и медицинское учреждение в другой стране, а также осуществляет запись на прием и оказывает помощь в организации поездки [7].

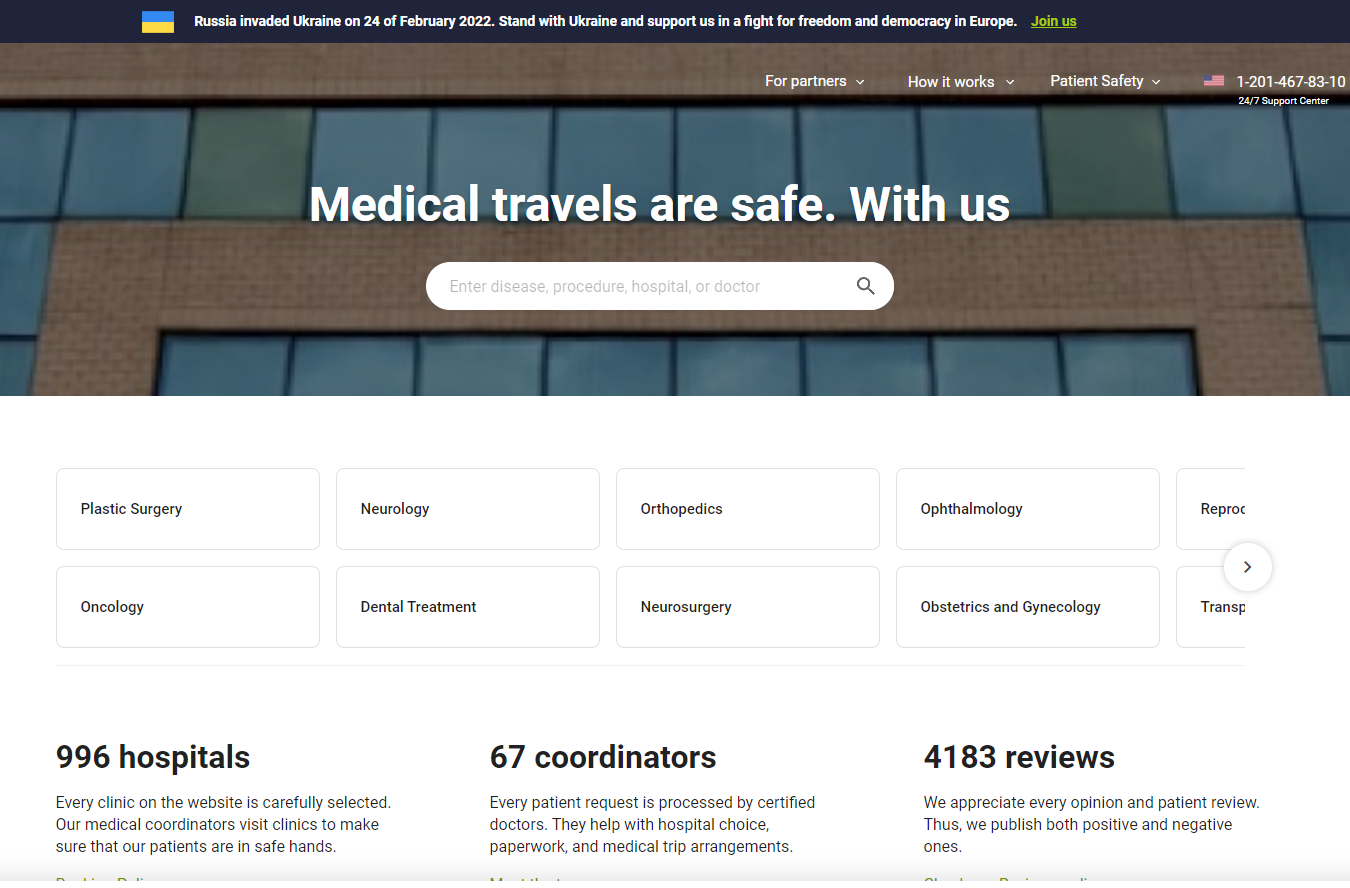


Рисунок 2.4 - Bookimed

Преимущества:

* Подбор врачей и медицинских учреждений по всему миру.
* Помощь в организации медицинской помощи за рубежом.
* Отзывы пациентов о врачах и медицинских учреждениях.
* Бесплатное использование сервиса для пациентов.

Недостатки:

* Ориентация на международную медицинскую помощь, а не на запись на прием к врачу в своем регионе.
* Возможность записи только к тем врачам, которые сотрудничают с платформой.

**Healthgorilla**

Ссылка: <https://www.healthgorilla.com/>

Healthgorilla – это американская платформа, предназначенная для обмена клинической информацией между врачами, пациентами и медицинскими учреждениями. Сервис предоставляет возможность записи на прием к врачу, доступ к медицинской документации и отправки результатов диагностических исследований между участниками медицинского процесса [8].

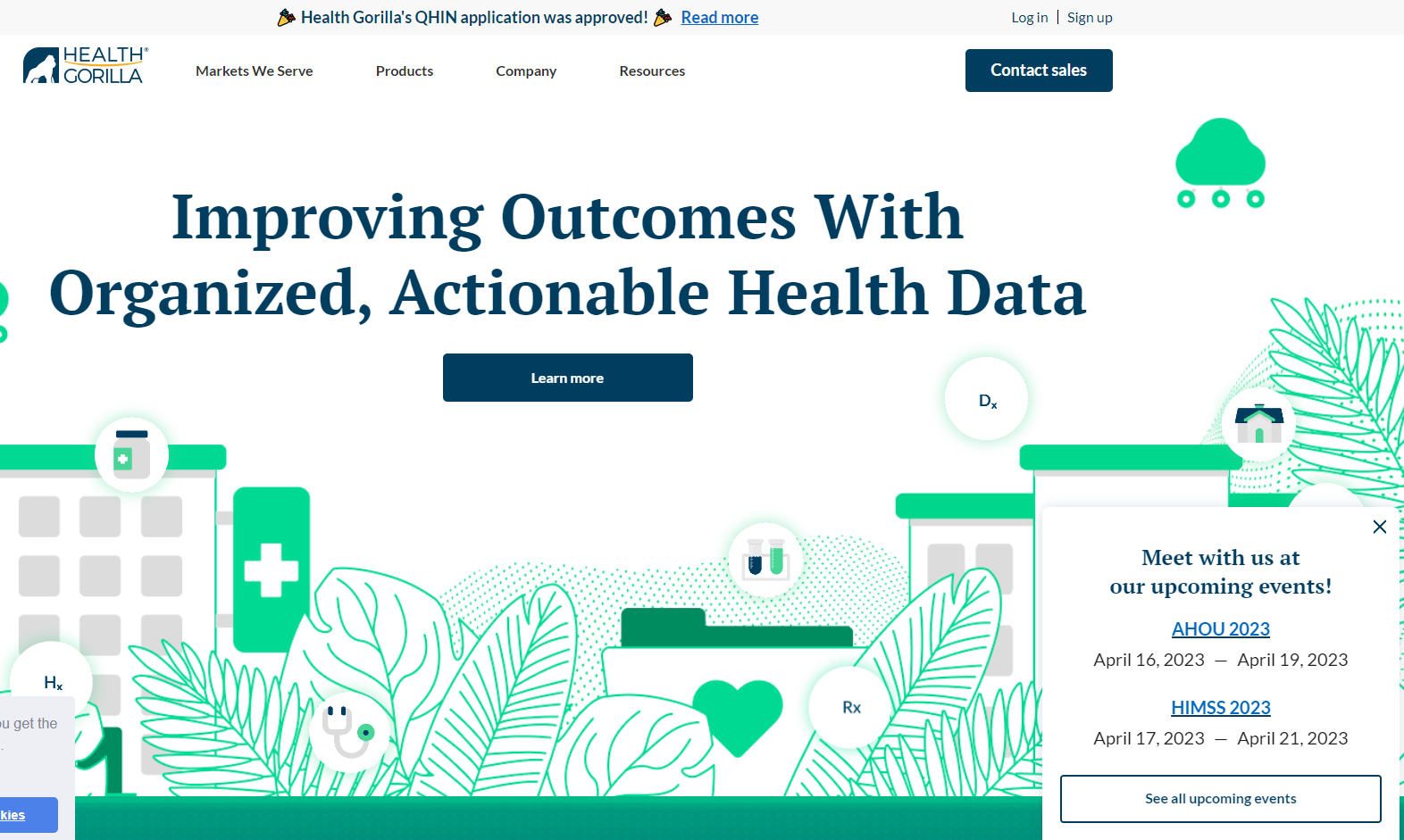


Рисунок 2.5 - Healthgorilla

Преимущества:

* Централизованный доступ к медицинской документации и результатам исследований.
* Упрощение коммуникации между врачами, пациентами и медицинскими учреждениями.
* Возможность записи на прием к врачу.
* Наличие мобильного приложения для iOS и Android.

Недостатки:

* Ограниченная география использования сервиса (в основном США).
* Возможность записи только к тем врачам, которые сотрудничают с платформой.

В заключение, существует множество информационных систем, предлагающих услуги по записи на прием к врачу. Все они имеют свои преимущества и недостатки, но стоит отметить, что выбор подходящей системы зависит от специфики медицинского учреждения и нужд пациентов. Однако, разработка новой информационной системы может предоставить дополнительные возможности и удовлетворить потребности рынка, предлагая уникальные функции и преимущества.

## 2.3 Обзор и обоснование выбора инструментальных средств

При разработке программного средства необходимо определиться с компонентами и технологиями, которые необходимы для реализации дипломного проекта.

Архитектура данного проекта – это веб-приложение. Архитектура приложения во многом предопределило и технологии, используемые для построения проекта.

Основные технологии, применяемые при создании приложения:

– серверный язык *JAVA*;

– язык запросов *SQL* (*H2*);

– фреймворк *Spring*;

– язык разметки *HTML*;

– таблица стилей *CSS*;

– клиентский язык *JavaScript*.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Язык Java появился в 1995 году – 90-е годы были вообще урожайными на новые языки и концепции программирования. В таком Эдеме языков важно было не заблудиться, по ошибке приняв за Священный Грааль технологию, которая не пройдет испытания временем. Java прошел испытания, хотя и очень долгие. Очень не рекомендуется путать этот язык с JavaScript – они по виду похожи, но это совсем разные языки.

Вероятно, в Java впервые реализовали концепцию того, что язык должен быть максимально изолирован от платформы разработки, чтобы применять его без изменений везде: в компьютерах, часах, сотовых телефонах, бытовой технике. С «железной частью» должна была справляться виртуальная машина (JVM), которая, собственно, и создавалась индивидуально под каждое устройство. Сам же язык был неизменен и в качестве результата выдавал байт-код. С самого начала было известно, что код не может исполняться очень быстро, но многие устройства не требовали высокой скорости исполнения. Кроме того, со временем появились оптимизирующие компиляторы, так что, в среднем, программа на Java работает раза в 2-3 медленнее, чем на C++. Постоянное сравнение с C/C++ здесь не случайно: многие современные языки взяли за основу его конструкции и синтаксис, так что, бывает, узнать сходу язык очень трудно. Вместе с тем, Java с тех пор сильно «размножилась», и даже J#, J и прочие аналоги являются не родными братьями, а лишь подобием [9].

Сама идея языка, вполне, кстати, достаточного для создания софта любой сложности, была сначала не понята: был ли, мол, смысл создавать между аппаратурой и кодом промежуточные слои исполняющих машин? Со временем сомнения рассеялись: появилась мультиязычная платфома .

NET, и даже в Windows появились слои – аппаратно-зависимые, платформо-независимые. Самое же простое объяснение – софт стал очень сложным, а программисты очень ленивыми, чтобы переписывать программы под каждый отдельный аппарат.

Но вернемся к языку. Как уже говорилось, чем-то он похож на C++, чем-то на старый добрый Бейсик. Нет сейчас ни одного языка, который бы не хвалился своими возможностями ООП, и Java здесь не отличается от канонов: классы и объекты здесь используются везде, даже в самых примитивных задачах вроде вывода строки на экран. Из особенностей можно отметить, что все объекты в языке создаются только динамически, а все функции являются методами классов. Множественное наследование не поддерживается, как в C++, как и «опасные» указатели. ООП дает много преимуществ, но и требует слишком многого – в случае Java памяти устройства никогда не будет слишком много. В остальном же, имеются библиотеки классов для практически всех задач; преимущественно – под написание клиентских и серверных приложений. Хозяин Java – Oracle – успешно использует язык для использования в разработках своей одноименной СУБД. На сегодняшний день язык считается наиболее востребованным на рынке.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET [10].

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.2.x.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Следующие две технологии – это *HTML* и *CSS*, которые предопределены архитектурой проекта. Имеется единственное обоснование их выбора – это веб-архитектура разрабатываемого программного средства.

*HTML* – язык гипертекстовой разметки, который используется для структурирования и отображения веб-страницы и ее контента. Для разработки данного программного средства будет применяться версия этого языка *HTML*5 [11].

*CSS* – это формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки *HTML.* Применяемая версия данной технологии – *CSS*3 [12].

Последняя из основных применяемый технологий – это *JavaScript*. С помощью данного языка программирования, можно создавать динамически обновляемый контент, управлять мультимедиа, анимировать изображения. Что касается обоснований выбора в качестве клиентского языка *JavaScript*, то здесь выбор достаточно очевиден, ведь конкурентов у данного языка в данной сфере практически нет. Лишь некоторые достоинства данного языка: возможность навигации и управления по *DOM HTML*-страницы, возможность управления браузером, гибкость подхода объектно-ориентированного программирования, поддержка асинхронности.

Для проектирования программного средства используется язык графического описания *UML*.

*UML* – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур [13].

*UML* является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML*-моделью. *UML* был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. *UML* не является языком программирования, на основании *UML*-моделей возможна генерация кода.

Следующим этапом данного пункта будет рассмотрение структуры программы посредствам следующих *UML*-диаграмм:

– диаграмма компонентов;

– диаграмма развёртывания;

– диаграмма последовательности.

Диаграмма компонентов – это элемент языка моделирования *UML*, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и другое.

Диаграмма компонентов для текущего проекта представлена на рисунке 4.1. Как видно, система состоит из трёх компонентов:

– СУБД *MySQL*;

– серверное приложение;

– браузер клиента.

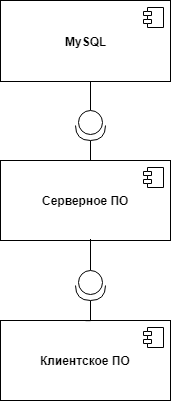


Рисунок 2.6 – Диаграмма компонентов

Каждый аспект отвечает за свой функционал. СУБД *MySQL* – за хранение и выборку данных, серверное приложение – за логику системы, а браузер клиента – за отображение информации.

# 3 Проектирование

## 3.1 Разработка архитектуры программного продукта

Чаще всего веб-приложения состоят как минимум из трёх основных компонентов:

Клиентская часть веб приложения — это графический интерфейс. Это то, что вы видите на странице. Графический интерфейс отображается в браузере. Пользователь взаимодействует с веб-приложением именно через браузер, кликая по ссылкам и кнопкам [14].

Серверная часть веб-приложения — это программа или скрипт на сервере, обрабатывающая запросы пользователя (точнее, запросы браузера). Чаще всего серверная часть веб-приложения программируется на PHP. При каждом переходе пользователя по ссылке браузер отправляет запрос к серверу. Сервер обрабатывает этот запрос, вызывая некоторый PHP-скрипт, который формирует веб-страничку, описанную языком HTML, и отсылает клиенту по сети. Браузер тут же отображает полученный результат в виде очередной веб-страницы.

База данных (БД, или система управления баазми данных, СУБД) - программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент. В случае форума или блога, хранимые в БД данные — это посты, комментарии, новости, и так далее. База данных располагается на сервере. Серверная часть веб-приложения (то есть, PHP скрипт) обращается к базе данных, извлекая данные, которые необходимы для формирования страницы, запрошенной пользователем.

Наше приложение также будет состоять из этих трех частей и основываться будет на Spring MVC. MVC — это не шаблон проекта, это конструкционный шаблон, который описывает способ построения структуры нашего приложения, сферы ответственности и взаимодействие каждой из частей в данной структуре.

Фреймворк Spring MVC обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними [15].

* Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).
* View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.
* Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Вся логика работы Spring MVC построена вокруг DispatcherServlet, который принимает и обрабатывает все HTTP-запросы (из UI) и ответы на них. Рабочий процесс обработки запроса DispatcherServlet'ом проиллюстрирован на следующей диаграмме:

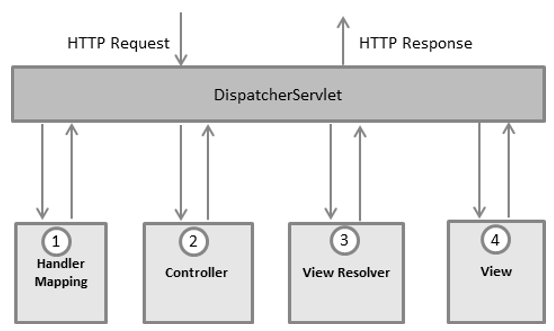


Рисунок 3.1 – Рабочий процесс обработки запроса

Ниже приведена последовательность событий, соответствующая входящему HTTP-запросу:

* После получения HTTP-запроса DispatcherServlet обращается к интерфейсу HandlerMapping, который определяет, какой Контроллер должен быть вызван, после чего, отправляет запрос в нужный Контроллер.
* Контроллер принимает запрос и вызывает соответствующий служебный метод, основанный на GET или POST. Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике, и возвращает в DispatcherServlet имя Вида (View).
* При помощи интерфейса ViewResolver DispatcherServlet определяет, какой Вид нужно использовать на основании полученного имени.
* После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере.

Все вышеупомянутые компоненты, а именно, HandlerMapping, Controller и ViewResolver, являются частями интерфейса WebApplicationContext extends ApplicationContext, с некоторыми дополнительными особенностями, необходимыми для создания web-приложений.

DispatcherServlet отправляет запрос контроллерам для выполнения определённых функций. Аннотация @Controllerannotation указывает, что конкретный класс является контроллером. Аннотация @RequestMapping используется для мапинга (связывания) с URL для всего класса или для конкретного метода обработчика.

Аннотация Controller определяет класс как Контроллер Spring MVC. В первом случае, @RequestMapping указывает, что все методы в данном Контроллере относятся к URL-адресу "/hello".

Следующая аннотация @RequestMapping(method = RequestMethod.GET) используется для объявления метода printHello() как дефолтного метода для обработки HTTP-запросов GET. Вы можете определить любой другой метод как обработчик всех POST-запросов по данному URL-адресу.

Вы можете написать вышеуказанный Контроллер по-другому, указав дополнительные атрибуты для аннотации @RequestMapping следующим образом

@RequestMapping(value = "/hello", method = RequestMethod.GET)  
Атрибут «value» указывает URL, с которым мы связываем данный метод (value = "/hello"), далее указывается, что этот метод будет обрабатывать GET-запросы (method = RequestMethod.GET). Также, нужно отметить важные моменты в отношении приведённого выше контроллера:

* Вы определяете бизнес-логику внутри связанного таким образом служебного метода. Из него Вы можете вызывать любые другие методы.
* Основываясь на заданной бизнес-логике, в рамках этого метода Вы создаёте Модель (Model). Вы можете добавлять аттрибуты Модели, которые будут добавлены в Вид (View). В примере выше мы создаём Модель с атрибутом «message».
* Данный служебный метод возвращает имя Вида в виде строки String. В данном случае, запрашиваемый Вид имеет имя «hello».

Spring MVC поддерживает множество типов Видов для различных технологий отображения страницы. В том числе — JSP, HTML, PDF, Excel, XML, Velocity templates, XSLT, JSON, каналы Atom и RSS, JasperReports и проч. Но чаще всего используются шаблоны JSP, написанные при помощи JSTL или HTML файлы, используя Thymeleaf.

На рисунке 3.2 представлена диаграмма последовательности, которая представляет собой один конкретный экземпляр работы программы под управлением пользователя.

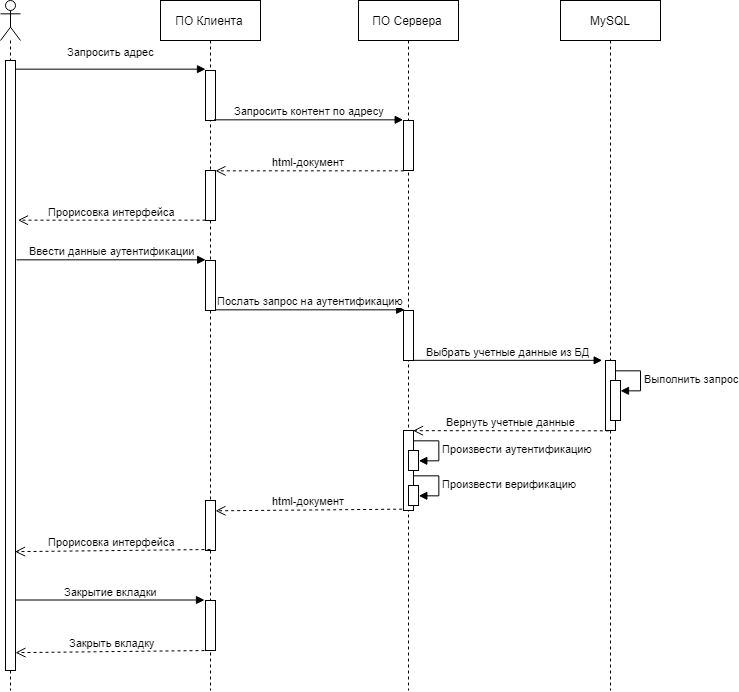


Рисунок 3.2 – Диаграмма последовательности

Диаграмма последовательности – это диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание – деятельность – уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках какого-либо определённого прецедента (отправка запросов и получение ответов).

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

На диаграмме последовательности изображен один актёр – пользователь системы, а также следующие объекты:

– ПО Клиента;

– ПО Сервера;

– MySQL.

Представлена последовательность действия в три этапа:

1. Пользователь вводит адрес сайта в браузере, а сервер отправляет клиенту html-документ.

2. Клиент вводит данные аутентификации. Браузер посылает запрос серверу на аутентификацию, а сервер посылает запрос на сервер MySQL, который выбирает данные и посылает их серверу. Сервер их преобразует и возвращает клиентскому браузеру. Браузер отображает информацию для клиента.

3. Клиент нажимает кнопку закрытия в браузере – вкладка закрывается. Работа системы завершена.

## 3.2 Проектирование структур хранения данных

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

На рисунке 3.3 представлена база данных, спроектированная по концептуальной модели.

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.

Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждое отношение должно обладать хотя бы одним ключом.

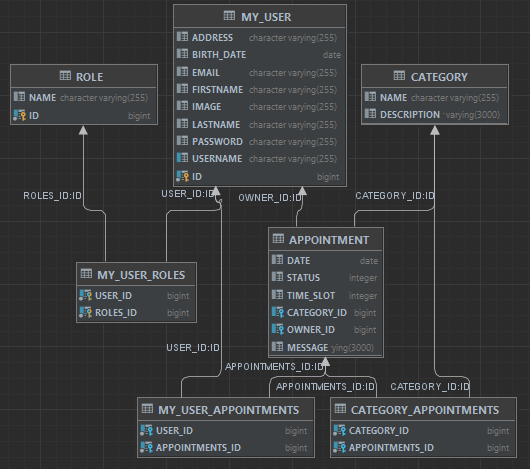


Рисунок 3.3– Спроектированная база данных

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации.

Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в 1НФ. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым, исключив из базы данных избыточность и аномалии обновления [16].

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой записи другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

На основе результатов выполненных действий была сформирована схема реляционной базы данных. На этой схеме отображены все отношения базы данных, а также связи между внешними и первичными ключами.

Таким образом, были определены ключевые атрибуты, связи между отношениями, организована целостность данных, нормализация некоторых отношений. Итоговым моментом стало построение нормализованной схемы разрабатываемой базы данных.

## 3.3 Описание реализации вариантов использования

Моделирование необходимо для понимания системы. Обычно, при этом единственной модели никогда не бывает достаточно. Наоборот, для понимания практически любой нетривиальной системы приходится разрабатывать большое количество взаимосвязанных моделей. В применении к программным системам это означает, что необходим язык, с помощью которого можно с различных точек зрения описать представления архитектуры системы на протяжении цикла ее разработки.

Диаграмма вариантов использования показывает, какая функциональность должна быть реализована в системе, основные функции, которые должны быть включены в систему, их окружение и взаимодействие функций с окружением.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне.

Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой [17].

Диаграмма вариантов использования (Use Case Diagram) представляет собой графическое представление возможностей и взаимодействий между различными акторами и системой. Она служит для описания функциональных требований к системе с точки зрения внешних пользователей. В данном случае, диаграмма вариантов использования для web-приложения "Магазин здоровой еды" будет выглядеть следующим образом:

1. Акторы:

* Гость;
* Пользователь (зарегистрированный пользователь);
* Менеджер;
* Администратор.

1. Варианты использования:

2.1. Гость:

* Просмотр главной страницы сайта;
* Просмотр информации о сайте;
* Просмотр списка категорий
* Регистрация в системе;
* Вход в систему;

2.2. Пользователь (наследует возможности Гостя):

* Создание записи к врачу;
* Просмотр истории записей;
* Редактирование профиля.

2.3. Менеджер:

* Просмотр списка записей;
* Изменение статуса записи;
* Изменение информации записи

2.4. Администратор:

* Просмотр списка пользователей/менеджеров;
* Добавление/изменение данных пользователей.

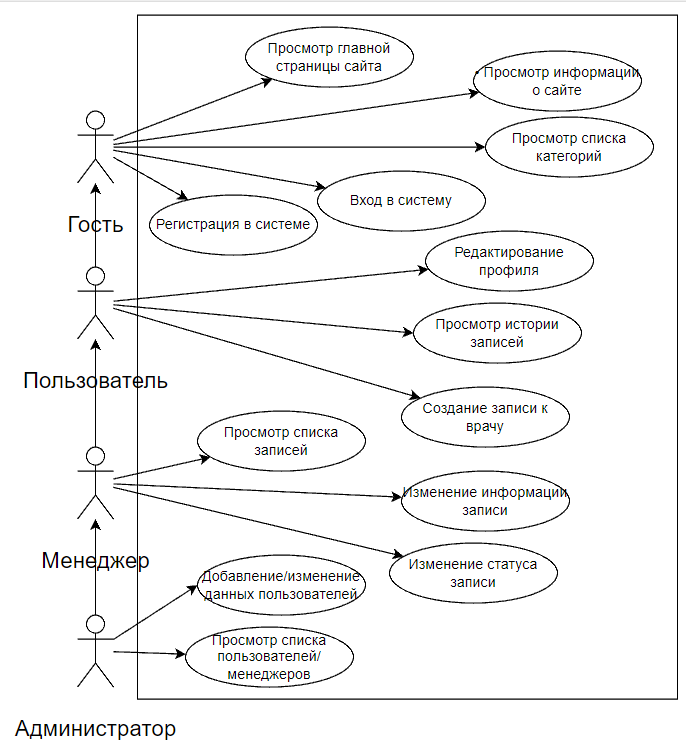


Рисунок 3.4 – Диаграмма вариантов использования

В результате анализа предметной области и постановки задачи были определены варианты использования системы. Для реализации данных вариантов необходимо определить взаимодействие частей проектируемого программного средства.

# 4 Реализация

## 4.1 Разработка классов информационной системы

Так как мы используем шаблон MVC, идет отдельно разработка классов модели (сущности), отдельно вида (страниц для браузера) и контроллеров (обработки действий пользователя).

Так как данные в шаблоне будут изменяться по требованию администратора ресурса, нам нужно создать класс-сущность для хранения этих данных.

Сущность (entity) – это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна.

Сущности уже были спроектированы раньше для базы данных. На их основе пишутся классы сервиса. Пример описания сущности приведен ниже.

Листинг 1. Сущность Appointment

import lombok.Data;

import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.ManyToOne;

import java.time.LocalDate;

@Entity

@Data

public class Appointment extends AbstractEntity{

@ManyToOne

private User owner;

@ManyToOne

private Category category;

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate date;

@Column(length = 3000)

private String message;

private AppointmentStatus status = AppointmentStatus.CREATED;

private TimeSlot timeSlot = TimeSlot.H09\_M00\_TO\_H09\_M30;

}

По описанию сущности автоматически будет сгенерирована таблица в базе данных, при ее отсутствии.

Сущность написана, переходим к созданию сервисов для доступа к бд. Сперва опишем интерфейс с основными командами.



Рисунок 4.1 – Интерфейс с основными командами

Затем переходим к написанию реализации этого интерфейса. На самом деле, реализация простая, мы вызываем соответствующие методы репозитория.

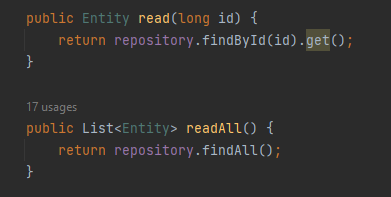


Рисунок 4.2 – Реализация интерфейса

Реализация есть, осталось связать нашу сущность шаблона с доступом к бд. Для этого наследуемся от только-что созданной реализации, и в конструктор передаем нашу сущность.

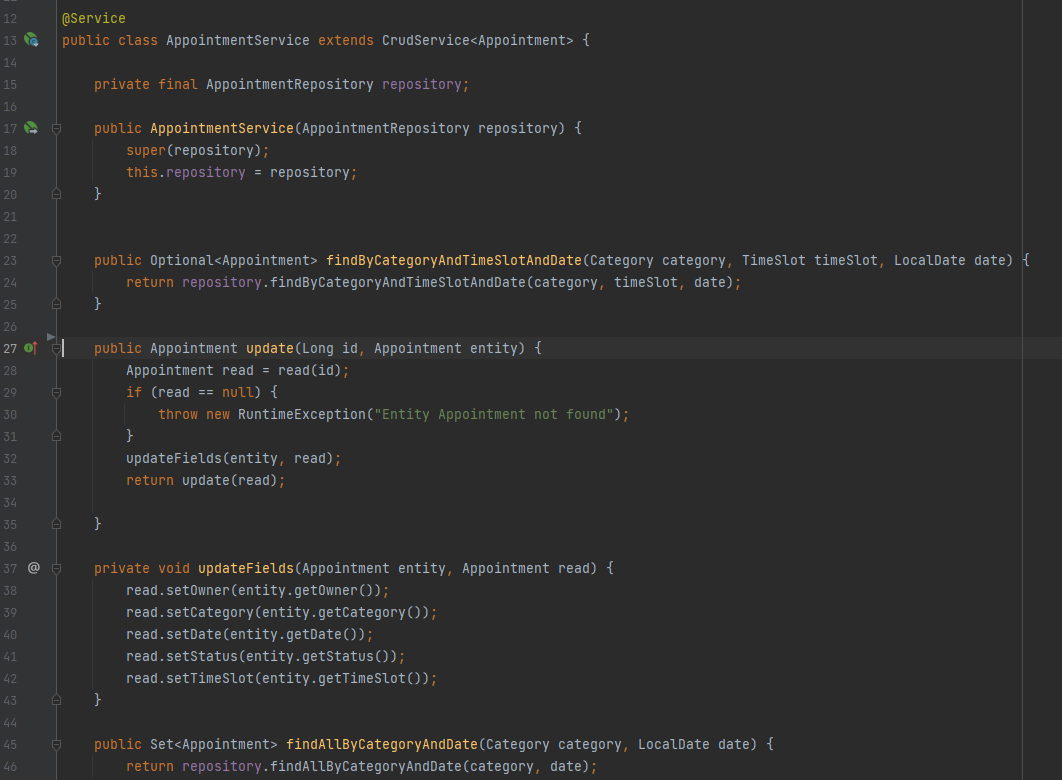


Рисунок 4.3 – Сервис доступа к таблице

Проделываем так для каждой сущности и готово, теперь, используя данный сервис, мы можем добавлять/редактировать/удалять данные в коде приложения.

Но это только в коде, пользователю это пока недоступно. Чтобы пользователь смог как-то взаимодействовать с приложением, создаем контроллеры страниц.

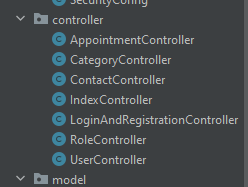


Рисунок 4.4 – Контроллеры страниц

По одному контроллеру для работы с соответствующими таблицами в базе данных.

Контроллер работает с данными таблиц базы данных, считывая или записывая в нее данные от пользователя. При запросе отображения, он возвращает страницу отображения. В этом контроллере реализованы методы создания, редактирования и удаления шаблонов по соответствующим адресам.

Листинг 2. Контроллер записей

package com.hescha.medicalappointment.controller;

import com.hescha.medicalappointment.model.Appointment;

import com.hescha.medicalappointment.model.AppointmentStatus;

import com.hescha.medicalappointment.model.Category;

import com.hescha.medicalappointment.model.SecurityService;

import com.hescha.medicalappointment.model.TimeSlot;

import com.hescha.medicalappointment.model.User;

import com.hescha.medicalappointment.service.AppointmentService;

import com.hescha.medicalappointment.service.CategoryService;

import com.hescha.medicalappointment.service.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.format.annotation.DateTimeFormat;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.access.annotation.Secured;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import org.springframework.stereotype.Controller;

import java.time.LocalDate;

import java.util.Arrays;

import java.util.HashSet;

import java.util.List;

import java.util.Optional;

import java.util.Set;

import java.util.stream.Collectors;

@Controller

@RequiredArgsConstructor

@RequestMapping(AppointmentController.CURRENT\_ADDRESS)

public class AppointmentController {

public static final String API\_PATH = "appointment";

public static final String CURRENT\_ADDRESS = "/" + API\_PATH;

public static final String MESSAGE = "message";

public static final String THYMELEAF\_TEMPLATE\_ALL\_ITEMS\_PAGE = API\_PATH;

public static final String THYMELEAF\_TEMPLATE\_ONE\_ITEM\_PAGE = THYMELEAF\_TEMPLATE\_ALL\_ITEMS\_PAGE + "-one";

public static final String THYMELEAF\_TEMPLATE\_EDIT\_PAGE = THYMELEAF\_TEMPLATE\_ALL\_ITEMS\_PAGE + "-edit";

public static final String REDIRECT\_TO\_ALL\_ITEMS = "redirect:" + CURRENT\_ADDRESS;

private final AppointmentService service;

private final UserService userService;

private final CategoryService categoryService;

private final SecurityService securityService;

@GetMapping

public String readAll(Model model) {

User loggedIn = securityService.getLoggedIn();

model.addAttribute("user", loggedIn);

model.addAttribute("list", service.readAll());

return THYMELEAF\_TEMPLATE\_ALL\_ITEMS\_PAGE;

}

@GetMapping("/{id}")

public String read(@PathVariable("id") Long id, Model model) {

model.addAttribute("entity", service.read(id));

return THYMELEAF\_TEMPLATE\_ONE\_ITEM\_PAGE;

}

@GetMapping("/{id}/changeStatus")

public String changeStatus(@PathVariable("id") Long id, AppointmentStatus status) {

Appointment read = service.read(id);

read.setStatus(status);

service.update(read);

return REDIRECT\_TO\_ALL\_ITEMS;

}

@GetMapping(path = {"/edit", "/edit/{id}"})

public String editPage(Model model, @PathVariable(name = "id", required = false) Long id) {

if (id == null) {

model.addAttribute("entity", new Appointment());

} else {

model.addAttribute("entity", service.read(id));

}

model.addAttribute("user\_list", userService.readAll());

model.addAttribute("category\_list", categoryService.readAll());

model.addAttribute("appointmentStatus\_list", AppointmentStatus.values());

model.addAttribute("timeSlot\_list", TimeSlot.values());

return THYMELEAF\_TEMPLATE\_EDIT\_PAGE;

}

@PreAuthorize("isAuthenticated()")

@PostMapping

public String save(@ModelAttribute Appointment entity, RedirectAttributes ra) {

if (entity.getId() == null) {

try {

User loggedIn = securityService.getLoggedIn();

entity.setOwner(loggedIn);

Appointment createdEntity = service.create(entity);

loggedIn.getAppointments().add(createdEntity);

userService.update(loggedIn);

ra.addFlashAttribute(MESSAGE, "Creating is successful");

return REDIRECT\_TO\_ALL\_ITEMS + "/" + createdEntity.getId();

} catch (Exception e) {

ra.addFlashAttribute(MESSAGE, "Creating failed");

e.printStackTrace();

}

return REDIRECT\_TO\_ALL\_ITEMS;

} else {

try {

service.update(entity.getId(), entity);

ra.addFlashAttribute(MESSAGE, "Editing is successful");

} catch (Exception e) {

e.printStackTrace();

ra.addFlashAttribute(MESSAGE, "Editing failed");

}

return REDIRECT\_TO\_ALL\_ITEMS + "/" + entity.getId();

}

}

@GetMapping("/{id}/delete")

public String delete(@PathVariable Long id, RedirectAttributes ra) {

try {

service.delete(id);

ra.addFlashAttribute(MESSAGE, "Removing is successful");

} catch (Exception e) {

e.printStackTrace();

ra.addFlashAttribute(MESSAGE, "Removing failed");

}

return REDIRECT\_TO\_ALL\_ITEMS;

}

@GetMapping("/check-availability")

public ResponseEntity<?> checkAvailability(@RequestParam("categoryId") Long categoryId,

@RequestParam("timeSlot") TimeSlot timeSlot,

@RequestParam("date")

@DateTimeFormat(iso = DateTimeFormat.ISO.DATE)

LocalDate date) {

Category category = categoryService.read(categoryId);

Optional<Appointment> existingAppointment = service.findByCategoryAndTimeSlotAndDate(category, timeSlot, date);

if (existingAppointment.isPresent()) {

return ResponseEntity.status(HttpStatus.CONFLICT).body("The selected TimeSlot and Category for the given date are already booked.");

} else {

return ResponseEntity.ok("The selected TimeSlot and Category for the given date are available.");

}

}

@GetMapping("/available-time-slots")

public ResponseEntity<List<TimeSlot>> getAvailableTimeSlots(@RequestParam("categoryId") Long categoryId,

@RequestParam("date") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date) {

Category category = categoryService.read(categoryId);

Set<Appointment> bookedAppointments = service.findAllByCategoryAndDate(category, date);

List<TimeSlot> availableTimeSlots = Arrays.stream(TimeSlot.values())

.collect(Collectors.toList());

for (Appointment appointment : bookedAppointments) {

availableTimeSlots.remove(appointment.getTimeSlot());

}

return ResponseEntity.ok(availableTimeSlots);

}

}

Класс наследует стандартный класс настройки безопасности. В нем мы сразу создаем данные для входа администратора и в методе конфигурации прописываем, какие пользователи куда могут попасть.

Поскольку использованные шаблоны нужно где-то хранить, создаем настройку подключения к базе данных. Все это делается в файле application.properties.

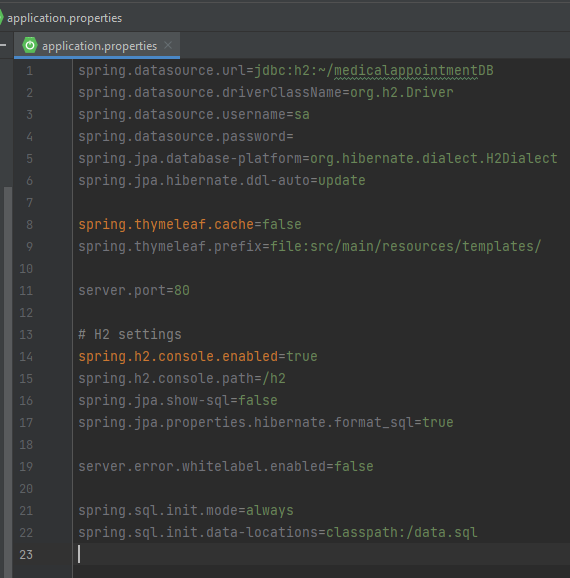


Рисунок 4.5 – Файл настройщик приложений

Проделав все эти операции, мы имеем следующую диаграмму классов

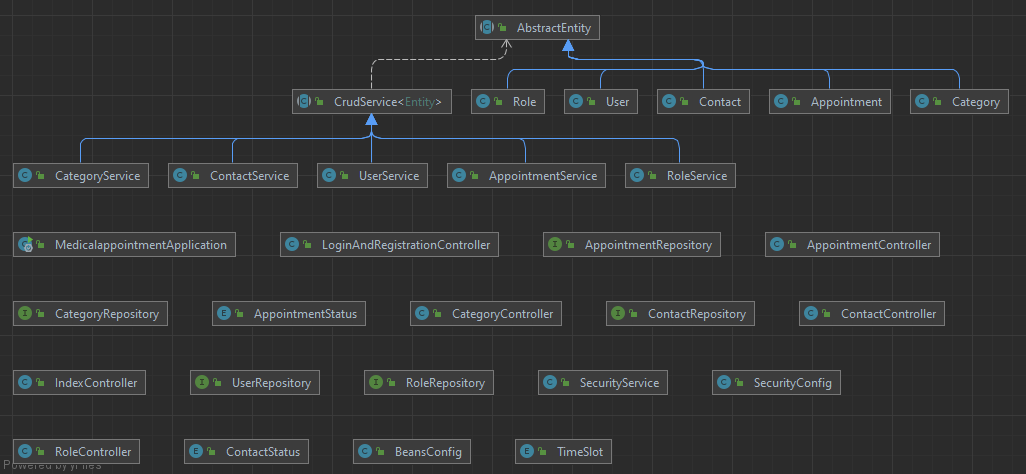


Рисунок 4.6 – Диаграмма классов

## 4.2 [Разработка интерфейса программного продукта](#_Toc474749001)

Определившись что и как должно быть, осталось спроектировать примерный пользовательский интерфейс. Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя и вычислительной системы. Важно помнить, что интерфейсы на стадии планирования и реализации могут отличатся, поскольку при написании приложения выявляется множество проблем и изменений.

ГОСТ «Эргономика взаимодействия человек-система», введенный в 2012 г., определяет пользовательский интерфейс (ПИ) как «компоненты интерактивной системы, предоставляющие пользователю информацию и являющиеся инструментами управления для выполнения определенных задач».

Проектирование пользовательского интерфейса – это создание тестовой версии приложения. Это начальный этап разработки пользовательского интерфейса, когда распределяются функции приложения по экранам, определяются макеты экранов, содержимое, элементы управления и их поведение.

На главной странице каждому пользователю доступна общая информация.

На этой же странице пользователь создает заявку на посещения врача в разделе «Appointment»

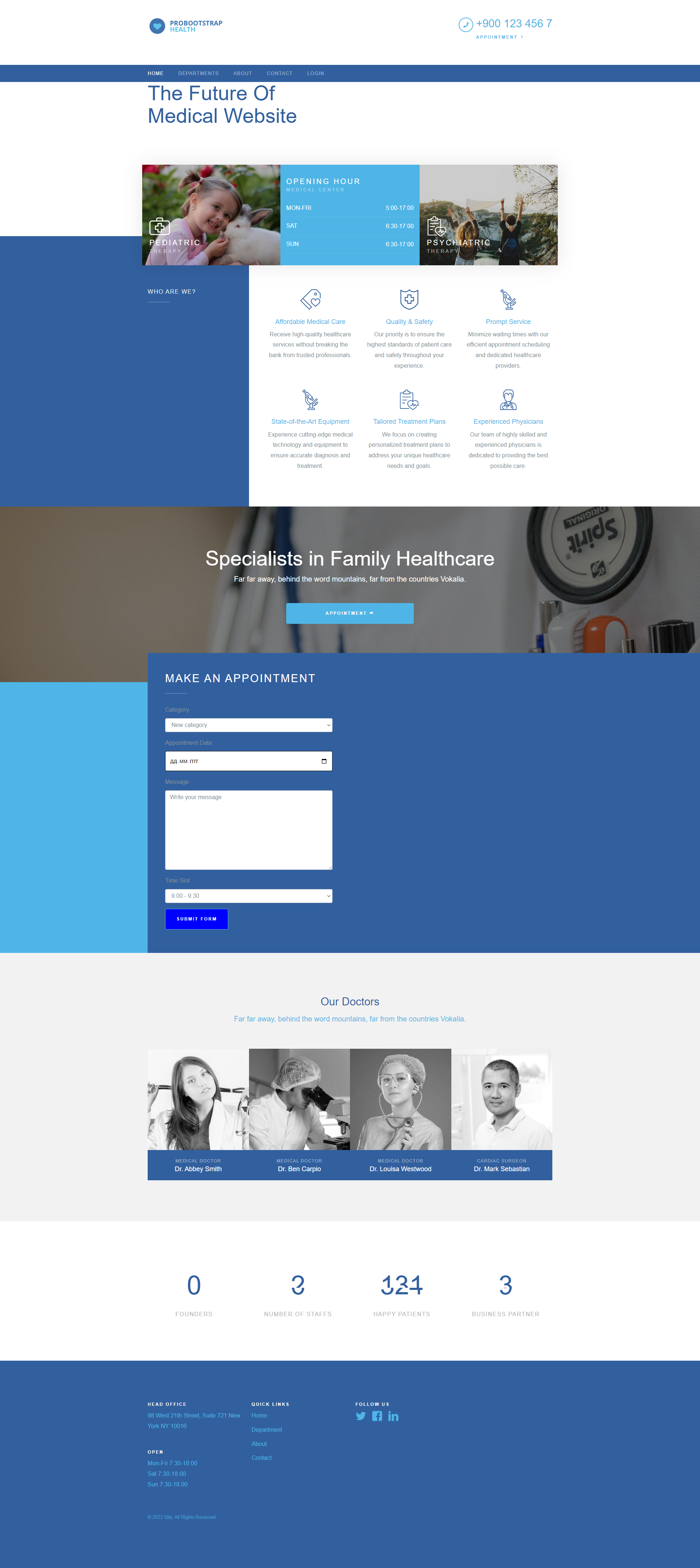


Рисунок 4.7 – Главная страница

На странице о нас можно увидеть более подробную информацию.

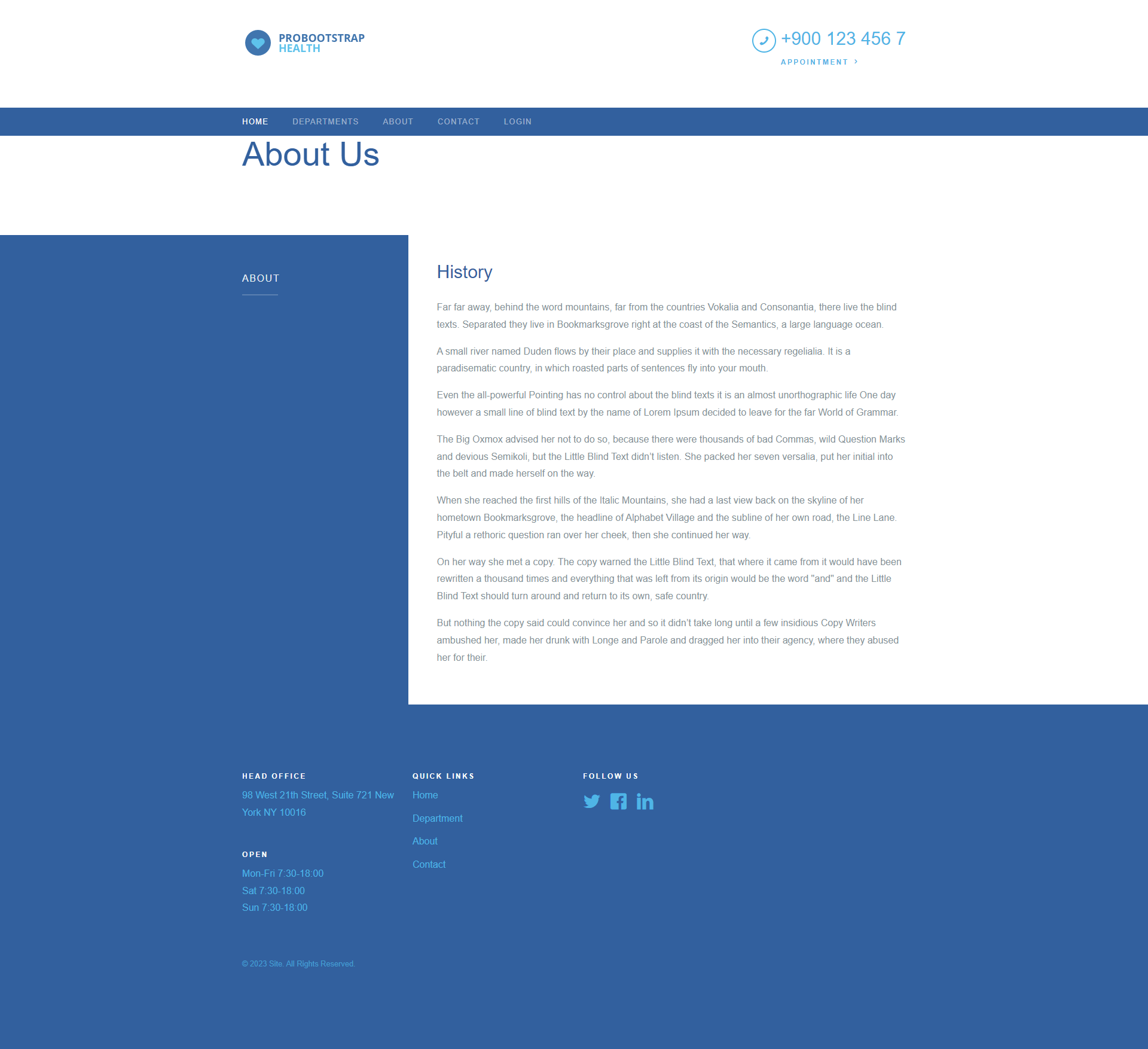


Рисунок 4.8 – Страница о нас

На странице департаментов (или проще категорий) можно посмотреть и почитать информацию обо всех категориях, на которые можно записаться.

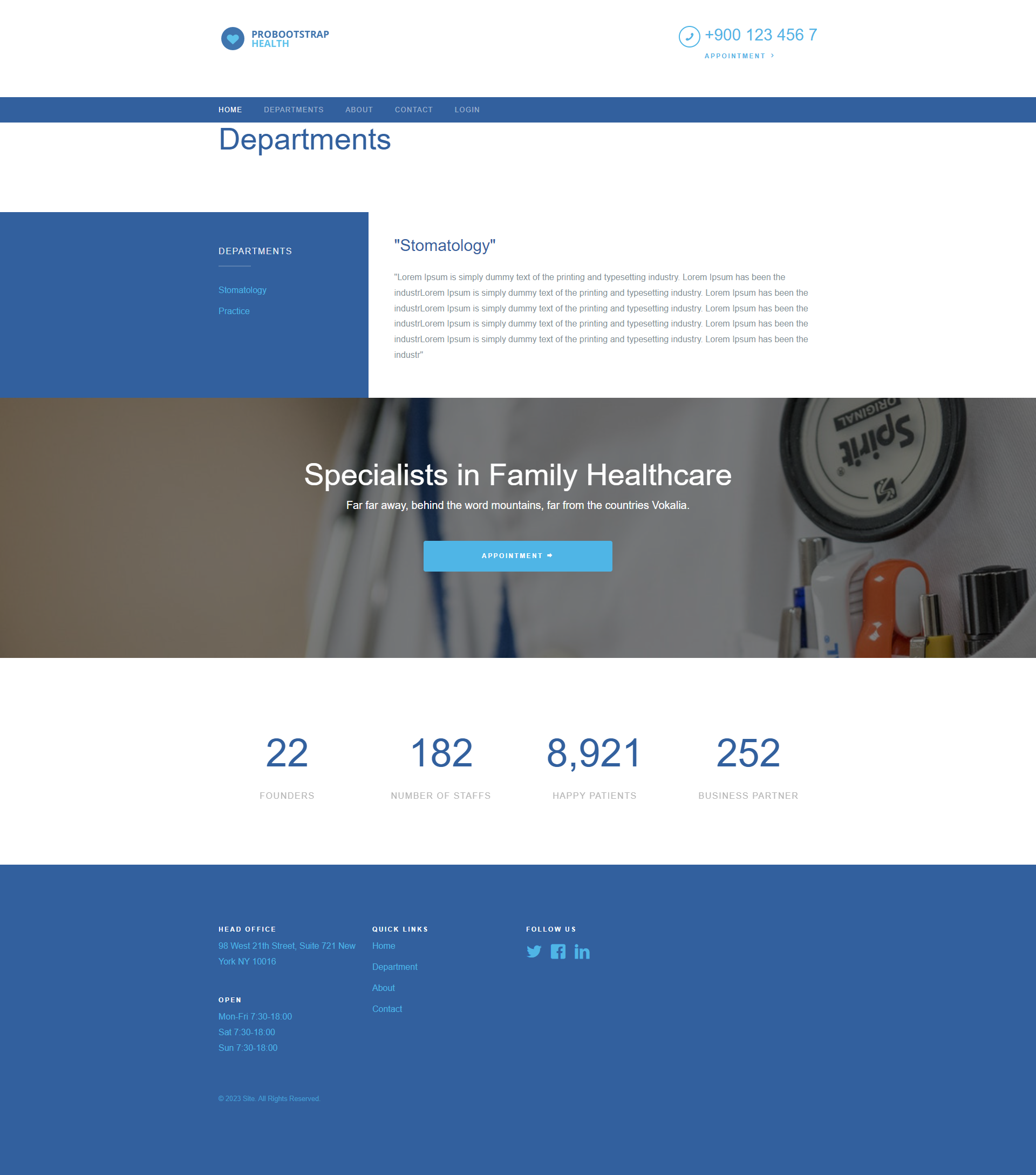


Рисунок 4.9 – Категории/Департаменты

Страница входа и регистрации для удобства объединены в одном месте.

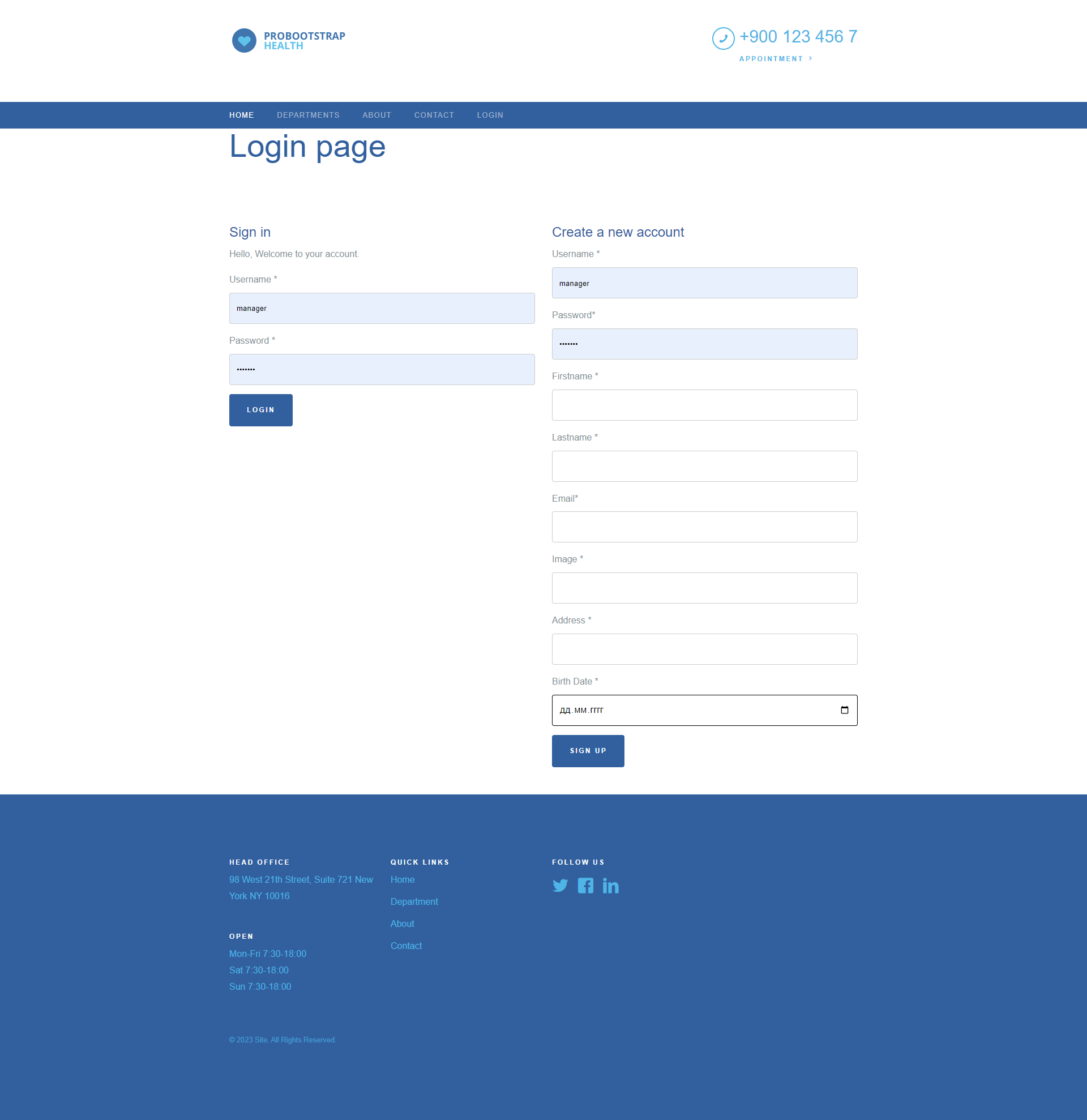


Рисунок 4.10 - Вход

Кроме того также существует страница контактов, на который каждый пользователь (даже незарегистрированный) может оставить свое пожелание, предложение или задать вопрос.

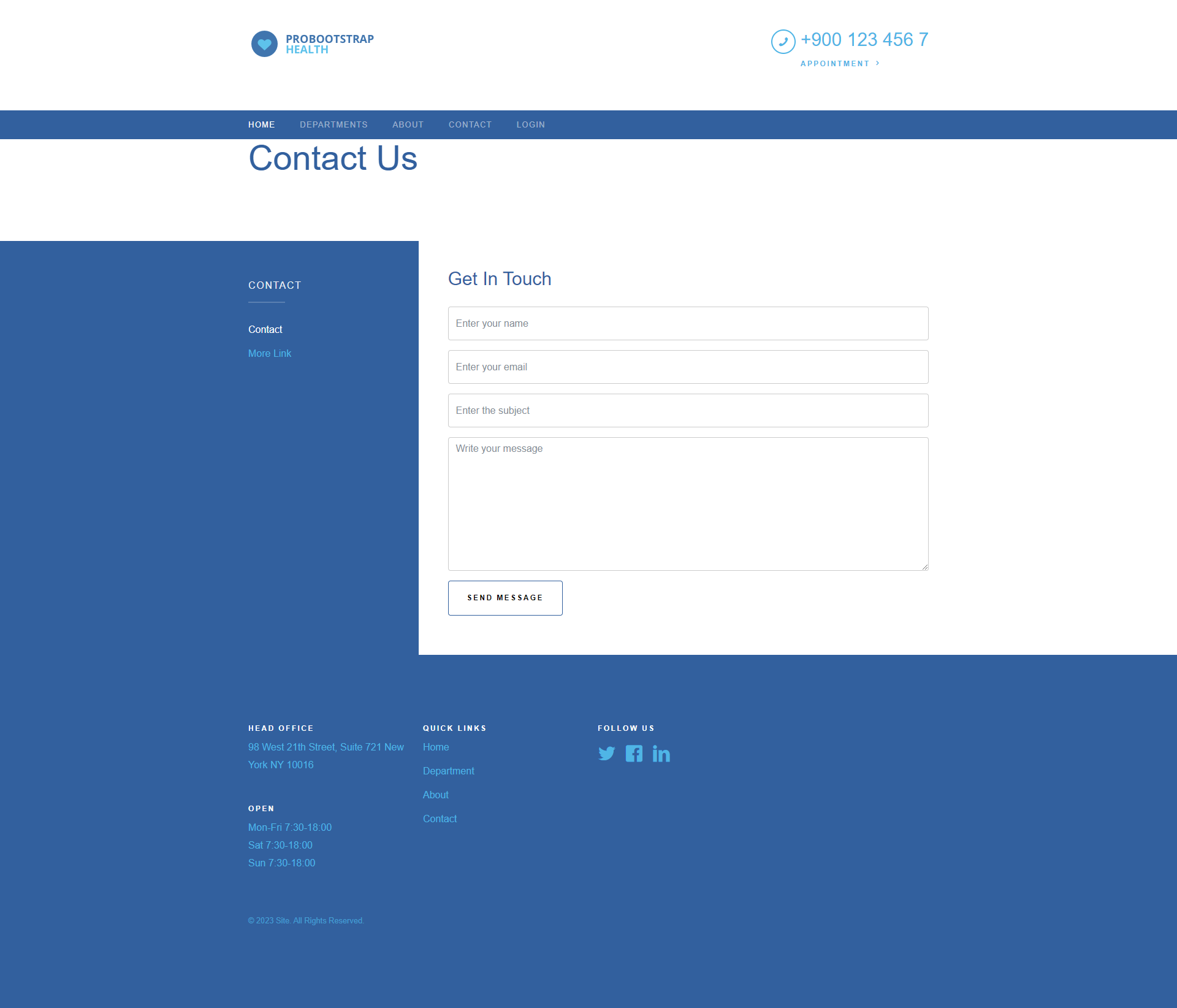


Рисунок 4.11 - Контакты

После входа в систему на странице записей пользователь может посмотреть свои предварительные запросы, которые сохранены но еще не отправлены для подтверждения.

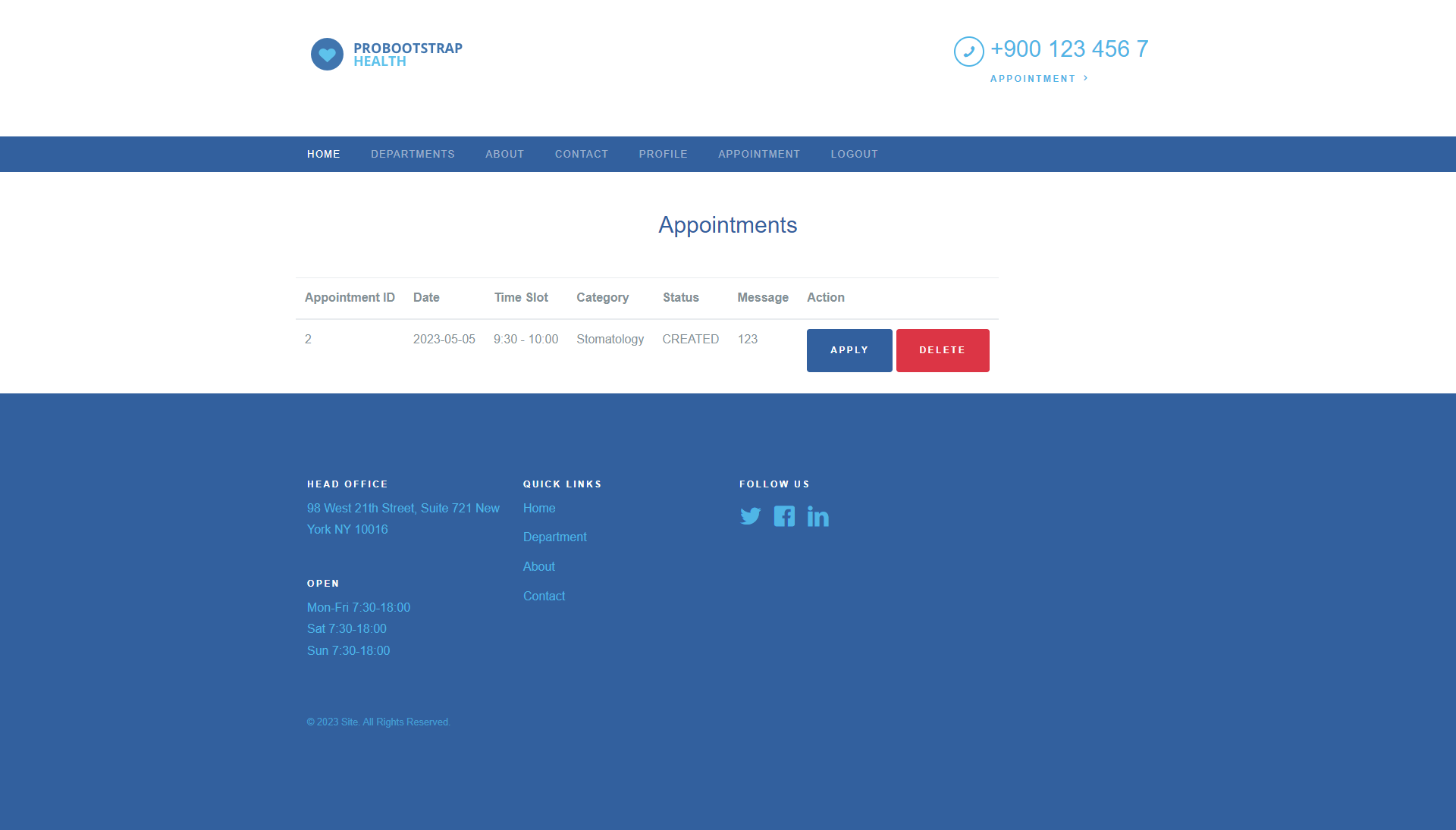


Рисунок 4.12 – Записи на прием

На странице входа и регистрации можно войти в свой аккаунт или зарегистрироваться соответственно.



Рисунок 4.13 – Успешная регистрация

После входа на сайт можно зайти на свой профиль и при необходимости изменить о себе информацию или просмотреть список своих записей к врачу.

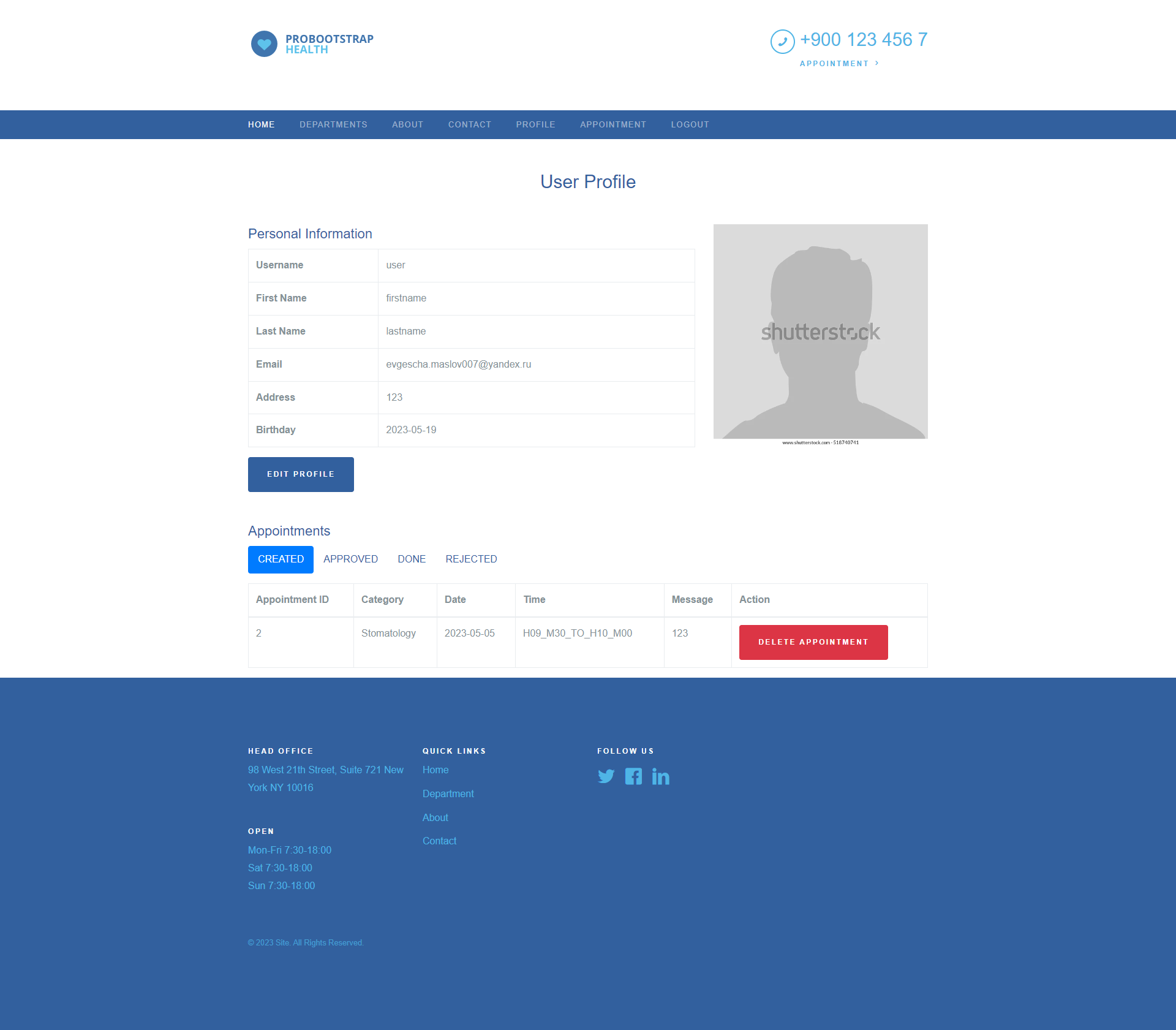


Рисунок 4.14 – Профиль пользователя

Сразу же после того как пользователь вошел в свой аккаунт, около каждого продукта будет заметна кнопка добавления товара в заказ.

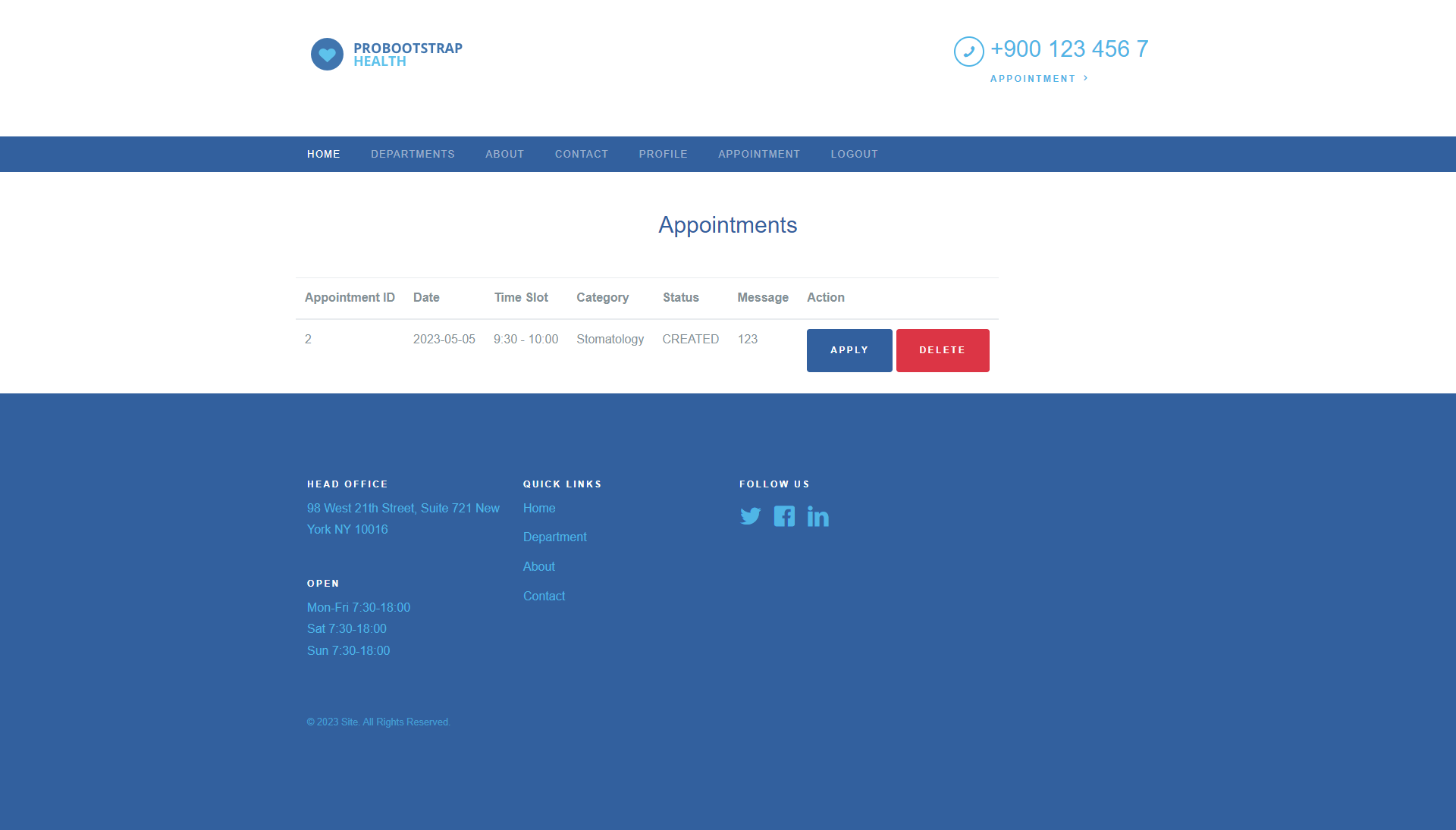


Рисунок 4.15 – Страница предсохраненных записей к врачу после авторизации

Администратору доступна страница категорий, на которой можно добавить новый или редактировать существующие категории.

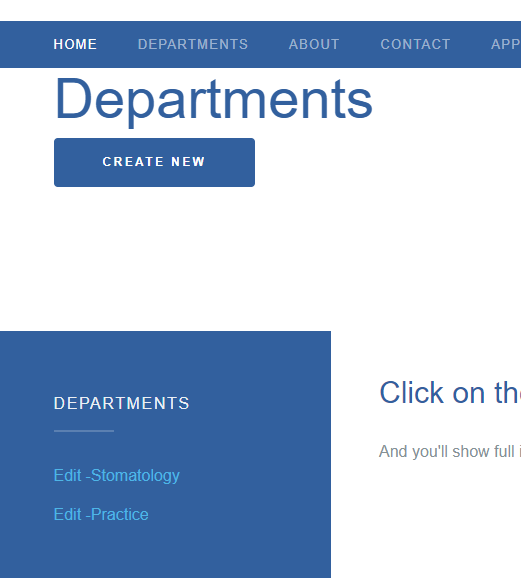


Рисунок 4.16 – Категории/Департаменты

На странице записей же администратору и менеджеру предоставлена функциональность для работы с записями клиентов.

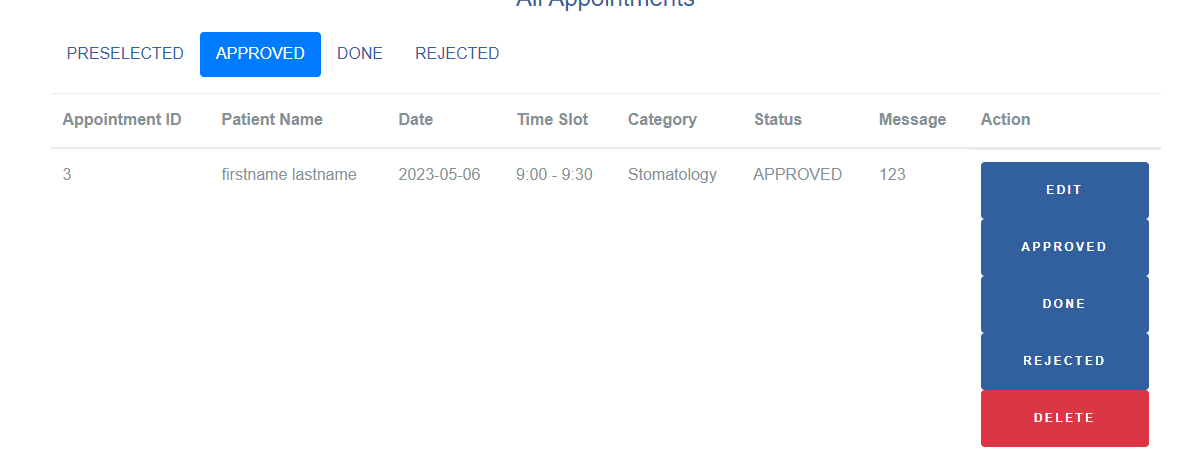


Рисунок 4.17 – Страница записей для администратора

На странице контактов же администратору будут отображены все активные запросы.

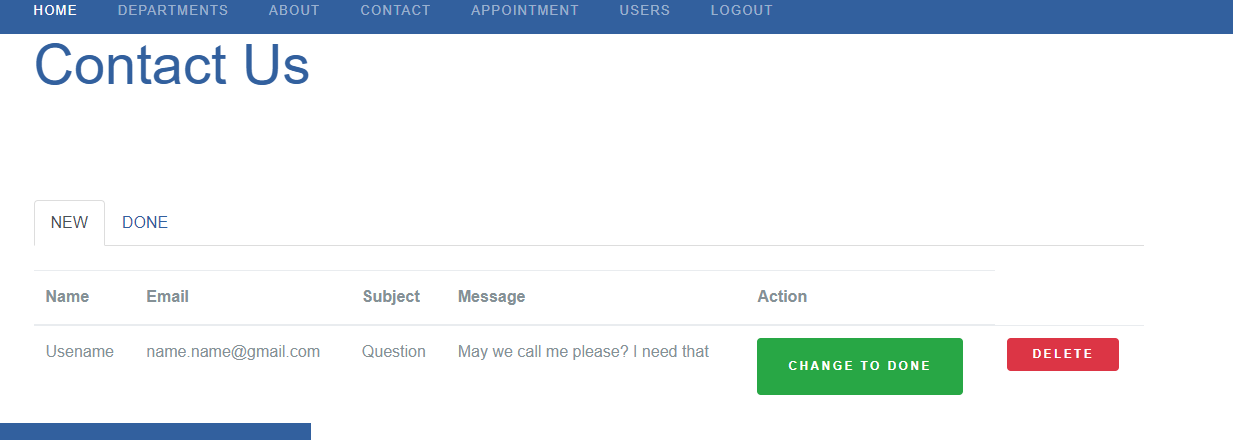


Рисунок 4.18 – Контакты

Стоит также отметить что администратору доступна страница всех пользователей, на которой он может изменять их данные и добавлять новых менеджеров в систему.

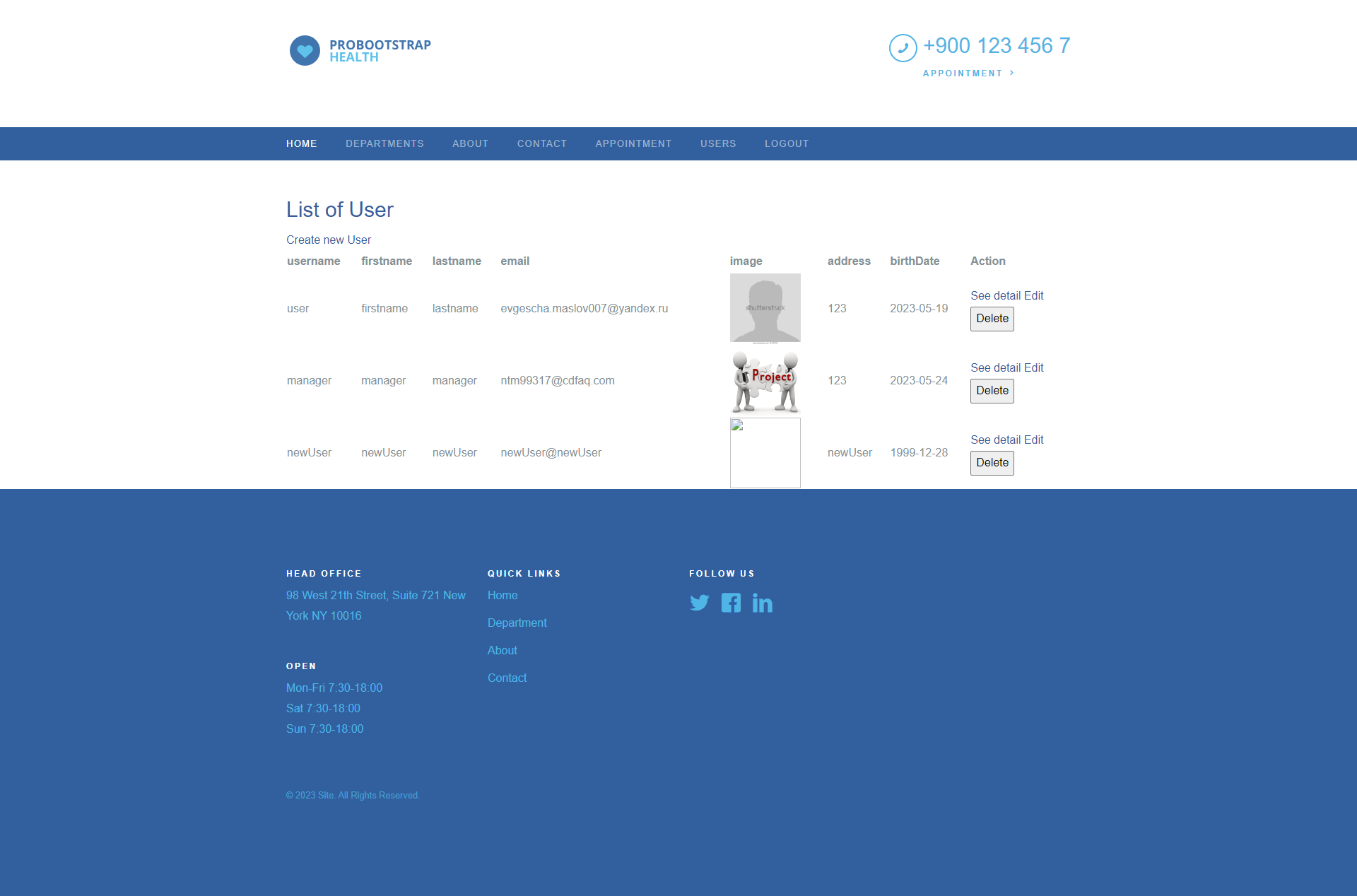


Рисунок 4.19 – Пользователи

## [Разработка алгоритмов реализации вариантов использования](#_Toc474749002)

При решении общей задачи, возникали более мелкие, для решения которых необходимо было использовать и разрабатывать алгоритмы. Несколько таких примеров будет показано ниже.

1. Сохранение в базе данных сущности, если ее еще там нет

@PreAuthorize("isAuthenticated()")

@PostMapping

public String save(@ModelAttribute Appointment entity, RedirectAttributes ra) {

if (entity.getId() == null) {

try {

User loggedIn = securityService.getLoggedIn();

entity.setOwner(loggedIn);

Appointment createdEntity = service.create(entity);

loggedIn.getAppointments().add(createdEntity);

userService.update(loggedIn);

ra.addFlashAttribute(MESSAGE, "Creating is successful");

return REDIRECT\_TO\_ALL\_ITEMS + "/" + createdEntity.getId();

} catch (Exception e) {

ra.addFlashAttribute(MESSAGE, "Creating failed");

e.printStackTrace();

}

return REDIRECT\_TO\_ALL\_ITEMS;

} else {

try {

service.update(entity.getId(), entity);

ra.addFlashAttribute(MESSAGE, "Editing is successful");

} catch (Exception e) {

e.printStackTrace();

ra.addFlashAttribute(MESSAGE, "Editing failed");

}

return REDIRECT\_TO\_ALL\_ITEMS + "/" + entity.getId();

}

}

1. Определение доступности слота для записи

<script th:inline="javascript">

document.querySelector('.probootstrap-form-appointment').addEventListener('submit', async (event) => {

event.preventDefault();

const categoryId = document.getElementById('category').value;

const date = document.getElementById('date').value;

const response = await fetch(`/appointment/available-time-slots?categoryId=${categoryId}&date=${date}`);

const availableTimeSlots = await response.json();

const selectedTimeSlot = document.getElementById('timeSlot').value;

if (availableTimeSlots.includes(selectedTimeSlot)) {

// Если выбранный таймслот доступен, отправьте форму

document.querySelector('.probootstrap-form-appointment').submit();

} else {

// Если выбранный таймслот недоступен, покажите список доступных таймслотов

showAvailableTimeSlots(availableTimeSlots);

}

});

function showAvailableTimeSlots(data) {

let result = "К сожалению данное время уже забронировано. \n" +

"Ниже список доступных слотов:\n\n";

data.forEach(slot => {

result += "\n" + timeSlotToReadableText(slot);

});

alert(result);

}

function timeSlotToReadableText(timeSlot) {

const timeSlotMapping = {

"H09\_M00\_TO\_H09\_M30": "9:00 - 9:30",

"H09\_M30\_TO\_H10\_M00": "9:30 - 10:00",

"H10\_M00\_TO\_H10\_M30": "10:00 - 10:30",

"H10\_M30\_TO\_H11\_M00": "10:30 - 11:00",

"H11\_M00\_TO\_H11\_M30": "11:00 - 11:30",

"H11\_M30\_TO\_H12\_M00": "11:30 - 12:00",

"H12\_M00\_TO\_H12\_M30": "12:00 - 12:30",

"H12\_M30\_TO\_H13\_M00": "12:30 - 13:00",

"H13\_M00\_TO\_H13\_M30": "13:00 - 13:30",

"H13\_M30\_TO\_H14\_M00": "13:30 - 14:00",

"H14\_M00\_TO\_H14\_M30": "14:00 - 14:30",

"H14\_M30\_TO\_H15\_M00": "14:30 - 15:00",

"H15\_M00\_TO\_H15\_M30": "15:00 - 15:30",

"H15\_M30\_TO\_H16\_M00": "15:30 - 16:00",

"H16\_M00\_TO\_H16\_M30": "16:00 - 16:30",

"H16\_M30\_TO\_H17\_M00": "16:30 - 17:00",

"H17\_M00\_TO\_H17\_M30": "17:00 - 17:30",

"H17\_M30\_TO\_H18\_M00": "17:30 - 18:00"

};

return timeSlotMapping[timeSlot] || "Unknown time slot";

}

</script>

## 4.3 Модульное тестирование алгоритмов реализации вариантов использования

Модульное тестирование, или юнит-тестирование - это процесс в программировании, который направлен на проверку отдельных небольших частей приложения, также называемых атомарными, которые можно исследовать изолированно от других подобных частей. При выполнении данного тестирования могут проверяться как отдельные функции или методы классов, так и сами классы, взаимодействие классов, небольшие библиотеки и отдельные части приложения. Довольно часто данный вид тестирования реализуется с использованием специальных технологий и инструментальных средств автоматизации тестирования, значительно упрощающих и ускоряющих разработку соответствующих тест-кейсов. Таким образом, оценивая каждый элемент отдельно и подтверждая правильность его работы, установить проблему значительно проще чем, если бы элемент был частью системы.

Примеры тестирования алгоритмов реализации вариантов использования:

1. Тестирование вывода продуктов (таблица 4.1).

Таблица 4.1 - Тестирование вывода продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| **Краткое описание** | **Предварительные условия** | **Шаги для воспроизведения** | **Ожидаемый результат** |
| Тестирование вывода категорий | Запустить программу  Добавить категорию | Перейти на страницу категорий | При переходе на страницу, будет отображен список категорий |
| Тестирование изменения категории | Запустить программу  Выбрать существующую категорию | Изменить данные  Сохранить изменения | Новые данные отображаются на странице |
| Тестирование удаление категории | Запустить программу  Выбрать категорию | Удалить выбранную категорию | Удаленная категория больше не отображается |

1. Тестирование записи к врачу (таблица 4.2).

Таблица 4.2 - Тестирование добавления брони

|  |  |  |  |
| --- | --- | --- | --- |
| **Краткое описание** | **Предварительные условия** | **Шаги для воспроизведения** | **Ожидаемый результат** |
| Добавление записи | Запустить программу  Перейти на страницу записи | Добавить запись | Новая запись отображается как в профиле, так и на странице записей |
| Удаление записи | Запустить программу  Перейти на страницу записи | Удалить запись | Запись больше не отображается на странице |
| Подтверждение записи | Запустить программу  Перейти на страницу записи | Нажать кнопку подтверждения записи | Статус записи изменился на подтвержденный |

# 5 Системное тестирование

Системное тестирование — это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Системное тестирование относится к методам тестирования чёрного ящика, и, тем самым, не требует знаний о внутреннем устройстве системы [18].

Тестирование на уровне системы представляет собой процесс проверки программного продукта в полностью интегрированной системе с целью убедиться в соответствии системы первоначальным требованиям. Этот вид тестирования относится к методам тестирования "черного ящика" и, таким образом, не требует знаний о внутренней структуре системы. Различают следующие виды тестирования:

* функциональное тестирование;
* проверка пользовательского интерфейса;
* тестирование совместимости;
* проверка безопасности;
* анализ производительности;
* автоматизированное тестирование;
* интеграционное тестирование.

Основными целями системного тестирования являются:

* демонстрация разработчикам и заказчикам соответствия программного обеспечения заявленным требованиям. Для заказчика это означает, что каждая желаемая функциональность, записанная в документе требований, прошла хотя бы один тест (обычно больше). В случае общедоступного ПО, проверено соответствие всех ключевых характеристик. Тест, соответствующий этой цели, называется валидационным (проверка достоверности). Успешная валидация говорит о корректной работе системы;
* выявление ситуаций, когда ПО демонстрирует некорректное, нежелательное или несоответствующее спецификации поведение. Таким образом, поиск ошибок направлен на устранение неправильной работы системы, такой как сбой системы, нежелательное взаимодействие с другими системами, неверные расчеты, повреждение данных. Тестирование, выполняющее эту задачу, называется тестированием дефектов (defect testing). Успешный тест в этом контексте указывает на наличие ошибки в системе, которую затем исправляют.

## 5.1 Функциональное тестирование

Функциональное тестирование — это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает.

Функциональные требования включают в себя:

* функциональная пригодность;
* точность;
* способность к взаимодействию;
* соответствие стандартам и правилам;
* защищённость.

Тестирование и отладка программы являются наиважнейшими этапами разработки любых программных продуктов. Цель этого этапа - проверка правильности и точности реализации функций, выполнение которых возлагается на данный программный продукт. В случае выявления некоторых неточностей и ошибок необходимо проведение работ по их исправлению и доработке программного продукта до требуемого уровня.

На основе функций, которые должны быть протестированы в разработанной информационной обучающей системе, составлен чек-лист (checklist), приведенный в таблице 5.1.

Таблица 5.1 - Чек-лист

|  |  |  |  |
| --- | --- | --- | --- |
| **Тестируемый модуль** | | **Тестируемая функция** | **Результат** |
| Вход в систему | | Вход в систему | Выполнено успешно |
| Выход из системы | | Выход из системы | Выполнено успешно |
| Просмотр категорий | | Отображение информации и зависимых списков | Выполнено успешно |
| Просмотр контактов | Отображение информации других таблиц на основе выбранного значения | | Выполнено успешно |
| Работа с записями | Добавление, редактирование и удаление данных с выводом информации подчиненной таблицы | | Выполнено успешно |
| Работа с пользователями | Добавление, редактирование и удаление данных с выводом информации подчиненной таблицы | | Выполнено успешно |

В качестве тестирования программного продукта был выбран тест-кейс (**Test Case**).

**Test Case** - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Тестирование производилось на ОС «Windows 10».

## 5.2 Оценка безопасности

Обеспечение безопасности информационных систем представляет собой ряд мероприятий, направленных на предотвращение несанкционированных воздействий на защищаемую информацию, а также её утечки. Поскольку приложение построено на базе фреймворка Spring, вопросы безопасности берет на себя Spring Security.

Spring Security это Java/JavaEE framework, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Проект был начат Беном Алексом (Ben Alex) в конце 2003 года под именем «Acegi Security», первый релиз вышел в 2004 году. Впоследствии проект был поглощён Spring'ом и стал его официальным дочерним проектом. Впервые публично представлен под новым именем Spring Security 2.0.0 в апреле 2008 года.

Ключевые объекты контекста Spring Security:

SecurityContextHolder, в нем содержится информация о текущем контексте безопасности приложения, который включает в себя подробную информацию о пользователе(Principal) работающем в настоящее время с приложением. По умолчанию SecurityContextHolder используетThreadLocal для хранения такой информации, что означает, что контекст безопасности всегда доступен для методов исполняющихся в том же самом потоке. Для того что бы изменить стратегию хранения этой информации можно воспользоваться статическим методом класса SecurityContextHolder.setStrategyName(String strategy). Более подробно SecurityContextHolder.

SecurityContext, содержит объект Authentication и в случае необходимости информацию системы безопасности, связанную с запросом от пользователя.

GrantedAuthority отражает разрешения выданные пользователю в масштабе всего приложения, такие разрешения (как правило называются «роли»), например ROLE\_ANONYMOUS, ROLE\_USER, ROLE\_ADMIN.

UserDetails предоставляет необходимую информацию для построения объекта Authentication из DAO объектов приложения или других источников данных системы безопасности. Объект UserDetailsсодержит имя пользователя, пароль, флаги: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled и Collection — прав (ролей) пользователя.

UserDetailsService, используется чтобы создать UserDetails объект путем реализации единственного метода этого интерфейса.

UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;

Позволяет получить из источника данных объект пользователя и сформировать из него объект UserDetails который будет использоваться контекстом Spring Security.

**5.3 Тестирование производительности**

Тестирование производительности**-** это тестирование, которое проводится для определения скорости работы системы или её части при заданной нагрузке. Тестирование производительности стремится учесть производительность уже на стадии проектирования и моделирования и системы, до начала основной стадии разработки.

Тестирование производительности служит таким типичным целям:

* для демонстрации того, что система удовлетворяет критериям производительности;
* для определения производительность какой из двух или нескольких систем лучше;
* для определения, какой элемент нагрузки или часть системы приводит к снижению производительности.
* Для оценки времени загрузки страниц системы был использован специализированный веб-сервис «https://developers.google.com».

Приложение запускается за 5 секунд:

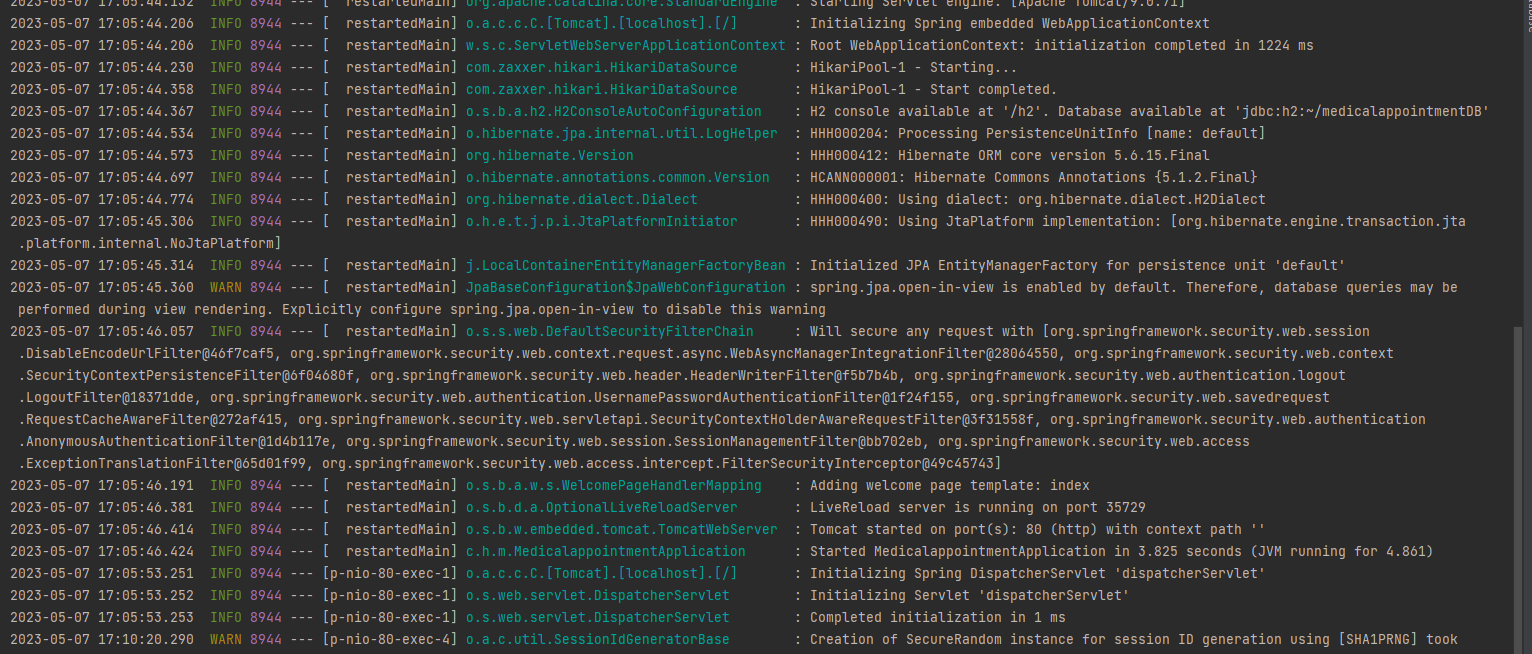


Рисунок 5.1 - Тестирование

# 6 Экономическая часть

## 6.1 Расчет трудоемкости разработки программного обеспечения

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).

## 6.2 Определение стоимости разработки программного обеспечения

Затраты времени на разработку ПО могут также определяться эмпирическим путем. В этом случае затраты времени могут включать:

- затраты труда на подготовку и описание задачи – *tоп*;

- затраты труда на исследование алгоритма решения задачи – *tис*;

- затраты труда на разработку алгоритма (блок-схем) – tал;

- затраты труда на программирование алгоритма по блок-схеме – *tпр*;

- затраты труда на отладку программы – *tотл*;

- затраты труда на подготовку документов по задаче состоят из затрат труда на подготовку рукописей и времени на оформление документов – *tд.*

Суммарные затраты труда рассчитываются как сумма составных затрат труда по формуле (6.1):

. (6.1)

Расчет суммарных затрат времени представлен в таблице 6.1.

Таблица 6.1 - Ориентировочное распределение затрат времени

|  |  |  |
| --- | --- | --- |
| Вид работ | Трудоемкость в часах | |
| Всего, человеко-часов | в том числе машинное время, машино-часов |
| Подготовку и описание задачи | 14 | - |
| Исследование алгоритма решения задачи | 30 | - |
| Разработка алгоритма | 32 | - |
| Программирование алгоритма | 240 | 240 |
| Отладка программы | 62 | 62 |
| Подготовка и оформление документов | 36 | 30 |
| Итого: | =414 | Σ*tмаш*=332 |

### 

6.2.1 Расчет затрат на разработку программного обеспечения

Затраты на оплату (*ЗОТ*) труда разработчика ПО включают затраты на оплату труда и отчисления от фонда заработной платы.

Затраты на оплату трударазработчика ПО рассчитывается в руб. по формуле (6.2):

(6.2)

где – суммарные затраты труда на разработку и сопровождение ПО, (таблица 6.1) ч.;

*КЧР* – среднемесячная расчетная норма рабочего времени (среднее количество часов работы в месяц), ч.;

– месячная заработная плата инженера-программиста, руб.

Согласно данным Министерства труда и социальной защиты Республики Беларусь среднемесячная расчетная норма рабочего времени при пятидневной рабочей неделе равна 169,3 ч.

Месячная заработная плата инженера-программиста включает:

а) оклад;

б) стимулирующие выплаты (надбавки и премии);

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

(6.3)

где – оклад работника, руб.;

– надбавка за работу в бюджетной организации, руб.;

– надбавка за стаж работы в бюджетной организации, руб.;

– надбавка за контрактную форму найма, руб.;

– ежемесячная премия, руб.

С 1 января 2020 года в Беларуси появилась новая система начисления зарплаты работникам бюджетной сферы. Правовое основание для этого – Указ Президента № 27 «Об оплате труда работников бюджетных организаций».

Вместо 27 тарифных разрядов для всех категорий трудящихся, вводится 18 разрядов для бюджетников. Основой для расчета будет не ставка 1 разряда, а базовая тарифная ставка, которая предполагается близкой по сумме к бюджету прожиточного минимума.

Базовая ставка – определяемая Правительством величина, на основании которой, через систему коэффициентов и доплат, будет формироваться заработная плата работников бюджетной сферы.

В соответствии с данным постановление постановлением расчет оплаты труда инженера-программиста, работающего в бюджетной организации, производится исходя из 4 разряда работ (тарифный коэффициент составляет 1,21). Базовая ставка принятая приказом на предприятии в году составляет 185 руб.

Оклад рассчитывается по формуле:

(6.4)

где БС – базовая ставка работников бюджетных организаций, руб.;

ТК – тарифный коэффициент, соответствующий разряду работ разработчика ПО.

Подставляя данные в формулу (6.4) получаем:

Стимулирующие выплаты:

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

(6.5)

Подставляя данные в формулу (6.5) получаем:

1. Надбавка за стаж работы в бюджетной организации при стаже работы до 5 лет устанавливается в размере 10% от базовой ставки.

(6.6)

Подставляя данные в формулу (6.6) получаем:

1. Надбавка за контрактную форму найма (до 40% от оклада):

(6.7)

Подставляя данные в формулу (6.7) получаем:

1. Премия ежемесячная (5% от оклада):

(6.8)

Подставляя данные в формулу (6.8) получаем:

На основе приведенных выше данных подставляя данные в формулу (6.3) мы получим месечную зароботную плату:

Рассчитаем затраты на оплату трударазработчика ПО по формуле (6.2):

Отчисления от фонда оплаты труда включают:

* отчисления в Фонд социальной защиты населения – 34% от ФЗП;
* страхование нанимателя от несчастных случаев на производстве – 0,6% от ФЗП.

Отчисления от фонда оплаты труда рассчитываются по формуле (6.9):

(6.9)

где – отчисления в Фонд социальной защиты населения (ставка отчислений составляет 34% от всех выплат работнику), руб.;

– страхование нанимателя от несчастных случаев на производстве и профзаболеваний (ставка отчислений составляет 0.6% от всех выплат работнику), руб.

Подставляя данные в формулу (6.9) получаем:

Затраты на оплату труда с учетом отчислений рассчитываются по формуле:

(6.10)

Подставляя данные в формулу (6.10) получаем:

6.2.2 Эксплуатационные затраты на оборудование

Стоимость оборудования хоть и не включается в себестоимость разработки программного обеспечения, но все же используется при расчете отдельных статей расходов. При написании программы в качестве оборудования предполагается использовать персональный компьютер, стоимость которого составляет: .

Суммарная стоимость эксплуатационных затрат рассчитывается по формуле (6.11):

(6.11)

где – годовые затраты на техническое обслуживание и ремонт оборудования, руб.;

– годовая стоимость электроэнергии, руб.;

– годовые амортизационные отчисления, руб.

Затраты на техническое обслуживание и ремонт (*СТО*) принимаются в размере 3-5% от стоимости оборудования, например:

(6.12)

Подставляя данные в формулу (6.12) получаем:

Амортизационные отчисления, процесс постепенного перенесения стоимости средств труда по мере их физического и морального износа на стоимость производимых с их помощью продукции в целях аккумуляции денежных средств для последующего полного восстановления. Годовые амортизационные отчисления производятся по установленным нормам амортизации, выражаются, в процентах к стоимости оборудования и рассчитываются по формуле (6.13):

(6.13)

где – стоимость компьютера;

– норма амортизации, которая рассчитывается по формуле (6.14):

(6.14)

где – нормативный срок службы (для персонального компьютера = 3 года).

Подставляя данные в формулы (6.14) и (6.13) получаем:

Годовая стоимость электроэнергии вычисляется по формуле (6.15):

(6.15)

где – мощность компьютера, КВт (примем равным 0.4 кВт);

– коэффициент загрузки, учитывающий использование оборудования по времени (0.9);

– стоимость 1 кВтч электроэнергии (0,27274 руб./кВтч для бюджетных организаций по состоянию 01.02.2020);

– коэффициент, учитывающий потери в сети (Kc=1,05);

– эффективный фонд рабочего времени, рассчитывается по формуле (6.16):

(6.16)

где = 255 – количество рабочих дней в 2020 году при пятидневной рабочей неделе (данные Министерства труда и соцзащиты РБ);

*d* = 7,97 – продолжительность рабочего дня, ч.;

*f* = 2% – планируемый процент времени на ремонт оборудования.

Подставляя данные в формулу (6.16) получаем:

Подставляя данные в формулу (6.15) получаем:

Однако, полученная стоимость эксплуатационных затрат – это значения годовых расходов. Необходимо их скорректировать в соответствии с временным коэффициентом (так как оборудование будет эксплуатироваться не весь год, а только в течение времени ), который определяется исходя из суммарных годовых эксплуатационных затрат, которые рассчитываются по формуле (6.17):

(6.17)

где – суммарная годовая стоимость эксплуатационных затрат, высчитываемая по формуле (6.18);

– эффективный фонд рабочего времени, (формула (6.16));

– общее время использования оборудования (из таблицы 6.1 – 343 ч.).

(6.18)

Подставляя данные в формулу (6.18) получаем:

Подставляя данные в формулу (6.17) получаем:

6.2.3 Затраты на материалы

Затраты на материалы включают расходы на бумагу, канцелярские принадлежности и другие материалы, используемые при разработке ПО.

Затраты на печать пояснительной записки к дипломному проекту составили – 15 бел. руб.

6.2.4 Накладные расходы

Накладные расходы, связанные с управлением, организационными расходами и прочими дополнительными затратами, составляют 30% от фонда заработной платы и вычисляются по формуле (6.19):

(6.19)

Подставляя данные в формулу (6.19) получаем:

Суммарные затраты на разработку программного обеспечения считаются как сумма фонда заработной платы и отчислений от него, эксплуатационных затрат, затрат на материалы, накладных расходов.

Себестоимость разработки программного обеспечения представлена в таблице 6.2.

Таблица 6.2 – Стоимость программного обеспечения

|  |  |  |
| --- | --- | --- |
| Наименование статей затрат | Обозначение | Сумма, руб. |
| 1. Затраты на материалы |  | 15 |
| 2. Эксплуатационные затраты |  | 70,6 |
| 3. Основная заработная плата |  | 1222,1 |
| 4. Отчисления от заработной платы |  | 422,8 |
| 5. Накладные расходы |  | 366,6 |
| 6. Полная себестоимость разработки и сопровождения ПО, *Сполн* | п.1+п.2+п.3+п.4+п.5 | 2097,1 |

### 6.3 Расчёт экономического эффекта от внедрения программного продукта

6.3.1 Экономический эффект у разработчика программного обеспечения

Заказчик оплачивает разработчику всю сумму расходов по проекту (полная себестоимость ПО из таблицы 6.2) с учетом прибыли разработчика и налога на добавленную стоимость с учетом качества, потребительских свойств продукции (ПО) и конъюнктуры рынка. Таким образом, в дипломном проекте отпускная цена программного обеспечения, представляет собой не цену за единицу продукции, а цену проекта вместе с его исходными кодами и документацией, за которую его можно продать и получить определенную выгоду. Прогнозируемая отпускная цена ПО (*ЦПО*) с учетом НДС рассчитывается по формуле (6.20):

, (6.20)

где *Сполн*– полная (плановая) себестоимость ПО, руб.,

*П* – прибыль разработчика ПО, руб.,

*СТНДС* – ставка налога на добавленную стоимость (=20%), в %.

Прибыль закладывается в цену исходя из уровня рентабельности (устанавливается студентом самостоятельно), расчет производится по формуле (6.21)

, (6.21)

где R– уровень рентабельности, % (в рамках дипломного проекта рекомендуемый уровень рентабельности ≈ 20%).

Подставляя данные в формулу (6.21) получаем:

Подставляя данные в формулу (6.20) получаем:

Ввиду того, 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 (6.22):

, (6.22)

где *СТП* – ставка налогообложения прибыли равная 18%.

Подставляя данные в формулу (6.22) получаем:

Таким образом, разработчик программного обеспечения может продать заказчику программное обеспечение по цене 3019 рублей 8 копеек с учетом НДС, что покроет затраты на разработку ПО, и обеспечит получение чистой прибыли от реализации в размере 343 рубля 9 копеек.

6.3.2 Экономический эффект от использования программного обеспечения у пользователя (заказчика)

Данный программный продукт отличается от аналогичных разработок конкурентов, следующими параметрами:

а) цена меньше, чем у конкурентов;

б) это универсальный программный продукт для любого пользователя;

в) хранения данных нас устройстве и отправка их в облако;

г) интуитивно понятный интерфейс.

Системные требования для данного программного продукта для ОС Windows:

- Windows 7/8/10;

- процессор – «Intel Pentium» (2 ядра по 2,0 ГГц) или аналог AMD;

- жёсткий диск – HDD 250 Гб;

- видеокарта – Nvidia или ATI Radeon 1024Мб DDR3;

- ОЗУ – DDR3 4000 Mб 1333 МГц.

Согласно различным источникам, текущая рыночная цена на подобный

программный продукт (лицензия на год) в Республике Беларусь колеблется в диапазоне от 3 000 бел. руб. до 4 000 бел. руб. Такая разбежка в ценовом диапазоне объясняется различным сроком полезного использования ПО (лицензия на год или на два).

Таким образом, рассчитанная отпускная цена на программный продукт, разрабатываемый в рамках данного дипломного проекта, является конкурентоспособной.

# 7 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) дается следующее определение понятию охраны труда:

Охрана труда - система обеспечения безопасности жизни и здоровья работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-противоэпидемические, лечебно- профилактические, реабилитационные и иные мероприятия и средства.

Требования по охране труда - нормативные предписания, направленные на сохранение жизни, здоровья и работоспособности работников в процессе их трудовой деятельности, содержащиеся в нормативных правовых актах, в том числе технических нормативных правовых актах.

Систему законодательных актов, регулирующих вопросы охраны труда в республике, 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. НПА, ТНПА, ЛНПА.

Охрана труда имеет большое социальное, экономическое и правовое значение.

Социальное значение охраны труда заключается в следующем:

* сохранение работоспособности и трудового долголетия работника;
* охрана жизни и здоровья работника от возможных воздействий вредных условий производства;
* охрана труда способствует гуманизации труда, содействует его культурно-техническому росту;

Экономическое значение охраны труда заключается в следующем:

* способствует росту производительности труда работников, росту производства и экономики;
* способствует экономии фонда социального страхования и сокращению потерь рабочего времени.

Работать с разработанным веб-сервисом планируется в жилом помещении, без предъявления каких-либо специальных требований. Специальной службы по охране труда не предусмотрено.

Рассмотрим характеристику объекта с точки зрения охраны труда на примере администратора веб-сервиса.

Проведем оценку факторов производственной среды, тяжести и напряжённости трудового процесса (таблицы 7.1-7.3).

Таблица 7.1 – Оценка факторов производственной среды

|  |  |  |
| --- | --- | --- |
| Факторы и показатели производственной среды | Гигиенические нормативы (ПДК, ПДУ) | Фактические величины |
| 1 | 2 | 3 |
| 2.4 Шум, дБА, дБ | 60 | 40 |
| 2.9 Электромагнитные поля и неионизирующие излучения |  |  |

Окончание таблицы 7.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 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 |
| 2.11 Микроклимат |  |  |
| 2.11.1 Температура воздуха, оС | 18-24 | 20 |
| 2.11.2 Относительная влажность, % | не более 60 | 50 |
| 2.11.3 Скорость движения воздуха, м/с | не более 0,3 | 0,1 |
| 2.12 Освещенность, лк | 300 | 560 |

Таблица 7.2 – Оценка тяжести трудового процесса

|  |  |
| --- | --- |
| Показатели тяжести трудового процесса | Фактическое значение показателя |
| 1 | 2 |
| 3.1 Физическая динамическая нагрузка, кгм |  |
| 3.1.1 Региональная нагрузка при перемещении груза на расстояние до 1 м | До 2 500 |
| 3.1.2 Общая нагрузка при перемещении груза на расстояние: - от 1 до 5 м | До 12 500 |
| - более 5 м |  |
| 3.2 Масса поднимаемого и перемещаемого груза вручную, кг | До 2 |
| 3.2.1 Подъем и перемещение тяжести при чередовании с другой работой | 3-12,5 |
| 3.2.2 Подъем и перемещение тяжестей постоянно в течение рабочей смены | 10 |

Окончание таблицы 7.2

|  |  |
| --- | --- |
| 1 | 2 |
| 3.2.3 Суммарная масса грузов, перемещаемых в течение каждого часа смены:  - с рабочей поверхности | До 350 |
| - с пола |  |
| 3.3 Стереотипные рабочие движения, количество за смену |  |
| 3.3.1 При локальной нагрузке | 12 000 |
| 3.3.2 При региональной нагрузке |  |
| 3.4 Статическая нагрузка, кг (силы) · с |  |
| 3.4.1 Одной рукой | До 36 000 |
| 3.4.2 Двумя руками | 20 000 |
| 3.4.3 С участием мышц корпуса, ног |  |
| 3.5 Рабочая поза (стоя) | Стоя 20 % |
| 3.6 Наклоны корпуса | 10 |
| 3.7 Перемещения в пространстве, обусловленные технологическим процессом, км |  |
| 3.7.1 По горизонтали | До 4 |
| 3.7.2 По вертикали |  |

Таблица 7.3 – Оценка напряженности трудового процесса

|  |  |
| --- | --- |
| Показатели напряженности трудового процесса | Характеристика показателей в соответствии с гигиеническими критериями |
| 1 | 2 |
| 4.1 Интеллектуальные нагрузки |  |
| 4.1.1 Содержание работы | Решение задач по инструкции |
| 4.1.2 Восприятие сигналов (информации) и их оценка | Восприятие сигналов, но не требуется коррекция действий |
| 4.1.3 Распределение функций по степени сложности задания | Обработка и выполнение задания |

Продолжение таблицы 7.3

|  |  |
| --- | --- |
| 1 | 2 |
| 4.1.4 Характер выполняемой работы | Работа по установленному регламенту |
| 4.2 Сенсорные нагрузки |  |
| 4.2.1 Длительность сосредоточенного наблюдения (в % от времени смены) | До 25 |
| 4.2.2 Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы | 60 |
| 4.2.3 Число производственных объектов одновременного наблюдения | 1 |
| 4.2.4 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% |
| 4.2.5 Работа с оптическими приборами (микроскопы, лупы и т.п.) при длительности сосредоточенного наблюдения (% времени смены) |  |
| 4.2.6 Наблюдение за экранами видеотерминалов (часов в смену): - при буквенно-цифровом типе отображения информации; | 5 |
| - при графическом типе отображения | До 3 |
| 4.2.7 Нагрузка на слуховой анализатор (при производственной необходимости восприятия речи или дифференцированных сигналов) | Разборчивость слов и сигналов от 75% до 50%. Помехи присутствуют |
| 4.2.8 Нагрузка на голосовой аппарат (суммарное количество часов, наговариваемое в неделю) |  |
| 4.3 Эмоциональные нагрузки |  |
| 4.3.1 Степень ответственности за результат собственной деятельности. Значимость ошибок | Ответственность за качество работы, влечёт дополнительные усилия со стороны руководства |

Окончание таблицы 7.3

|  |  |
| --- | --- |
| 1 | 2 |
| 4.3.2 Степень риска для собственной жизни | Исключена |
| 4.3.3 Степень ответственности за безопасность других лиц | Исключена |
| 4.4 Монотонность нагрузок |  |
| 4.4.1 Число элементов (приемов), необходимых для реализации простого задания или в многократно повторяющихся операциях | 8 |
| 4.4.2 Продолжительность выполнения простых производственных заданий или повторяющихся операций, с | 25-100 |
| 4.4.3. монотонность производственной обстановки (время пассивного наблюдения за ходом техпроцесса в % от времени смены) | 76-80 |
| 4.5 Режим работы |  |
| 4.5.1 Сменность работы | Односменная |

Из таблиц 7.2 – 7.3 следует, что администратор веб-сервиса подвержен всем видам нагрузок: эмоциональной, сенсорной и интеллектуальной.

На основании представленных данных разработаем карту рисков рабочего места администратора.

Для оценки рисков применяем классический метод. Оценка рисков рассчитывается по формуле (7.1):

*R = P × S,* (7.1)

где R – риск, балл; P – вероятность возникновения опасности, балл; S – серьезность последствий воздействия опасности, балл.

Исходя из значений P и S, определяем категорию риска. Категории рисков подразделяются на следующие: низкие (R < 6); умеренные (6 ≤ R ≤ 12); высокие (R > 12). Риски, отнесенные к категории «низкие» считаются допустимыми и управляемыми 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, отнесенные к категориям «умеренные» и «высокие» считаются недопустимыми и требуют разработки мер по управлению ими. Карта опасностей и рисков представлена в таблице 7.4.

Таблица 7.4 – Карта управления (умеренными) рисками

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Профессия, должность | Вид деятельности | Идентификационная  опасность | Серьёзность послед­ствий возникновения опасности, S | Вероятность возник­новения опасности, Р | Риск, R | Осуществляемые меры управления | Рекомендуе-мые действия | Срок исполнения |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Администратор веб-сервиса   * Дана характеристика объекта с точки зрения охраны труда: условия труда администратора интернет-магазина 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 интернет-магазина. * Произведена оценка организации охраны труда, производственной санитарии, промышленной и пожарной безопасности.   Администратор веб-сервиса | трудовая | Нервно-психические перегрузки | 2 | 4 | 8 | Инструкция по охране труда при работе с персональными электронно-вычислительными машинами | Самоконтроль | постоянно |
| 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 |

Окончание таблицы 7.4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Администратор веб-сервиса | трудовая | 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 в таблице 7.5.

Таблица 7.5 – Характеристика производственной санитарии и промышленной безопасности

|  |  |
| --- | --- |
| Исходные параметры | Характеристика реа­лизуемого парамет­ра |
| 1 | 2 |
| Организационные мероприятия по обеспечению охраны труда | - |
| Количество имевших место за отчетный период: | - |
| - аварий/количество пострадавших | - |
| - инцидентов/количество пострадавших | - |
| - несчастных случаев/количество пострадавших | - |

Окончание таблицы 7.5

|  |  |
| --- | --- |
| 1 | 2 |
| Технические средства и оборудование, обеспечивающие параметры микрокли­мата: | |
| - предусматриваемые системы вентиляции | естественная |
| - система отопления в помещении | Централизованное (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 |  |
| Технические средства и оборудование, обеспечивающие техническую безопас­ность: | |
| - знаки безопасности на оборудовании | - |
| - 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 |
| - 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, представленной в таблице 7.5 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».

Далее приведен расчёт необходимого количества светильников для освещения помещения методом светового потока.

Расчет искусственного освещения в цехе производится методом светового потока по формуле (7.2):

 (7.2)

где N – число светильников, обеспечивающее требуемую освещенность в помещении, шт.;

– нормируемая освещенность, лк (для III разряда зрительной работы и малого, среднего и большого контраста объекта с фоном – 300 лк);

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

S – площадь помещения, м2 (17,3 м2);

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

z – поправочный коэффициент, учитывающий неравномерность освещенности в помещении (примем равным 1,2);

– коэффициент использования светового потока, зависит от типа светильника, индекса помещения i, коэффициентов ρп, ρст, и ρр отражения потолка, стен и рабочей поверхности (в формулу значение коэффициента нужно подставлять в долях единицы).

Индекс помещения определяется по формуле (7.3):

 (7.3)

где a и b – длина и ширина помещения, м (для рассматриваемого помещения – 2,88 и 4 м);

hp – высота подвеса светильников, м (2,2 м).

Коэффициент отражения побеленных потолков принимается равным   
ρп= 50 %, стен, покрытых на высоту 1,8 м глазурованной плиткой,   
ρст = 50…70 %. Коэффициент отражения стен и потолка ξ зависит от характера отражающей поверхности: учитывая, что в помещении побеленные стены при не завешанных окнах и светлый деревянный потолок – ξ = 50 %;

Подставляя данные в формулу (7.3) получаем:

.

При данном индексе площади помещения и коэффициенте отражения стен и потолка ξ (50 %), коэффициент использования светового потока для светодиодных светильников η составляет 18. Подставляя данные в формулу (7.2), получаем необходимое количество светильников:



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 дирекции.

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

В таблице 7.6. отражены основные характеристики организации по степени подверженности пожарам.

Таблица 7.6 – 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 удаленного рабочего места до эвакуационного выхода, м | 20 |
| Количество эвакуационных выходов, шт. | 2 |
| Автоматические установки огнетушения | – |
| Тип извещателей о пожаре | дымовой |
| Первичные средства огнетушения | – |

Мероприятия по обеспечению пожарной безопасности соответствуют требованиям Декрета № 7, ТНПА противопожарного 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 и Инструк­цией по оценке условий труда при аттестации рабочих мест по условиям труда и предоставлению компенсаций по ее результатам.

В основу аттестации рабочих мест положены гигиенические критерии оценки условий труда, установленные в Санитарных нормах, правилах и гигие­нических нормативах «Гигиеническая классификация условий тру­да», утвержденных Постановлением Министерства здравоохранения Республи­ки Беларусь от 28.12.2012 г. № 211.

В соответствии с этим документом 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 7.7.

Таблица 7.7 – 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 (2020) | 1 (по контракту) |
| – женщин | 57 |
| – мужчин | 62 |
| Обеспечение ЛПП |  |
| Спецодеждой | – |
| Спецобувью | – |
| Средствами индивидуальной защиты органов зрения и дыхания | – |
| Средства обеззараживания кожи | вода, мыло, антисептик |
| Метод обеззараживания кожи | мытье рук |
| Периодичность медосмотра | 1 р. в 2 года |

В ходе выполнения раздела «Охрана труда» 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.
* Произведена оценка организации охраны труда, производственной санитарии, промышленной и пожарной безопасности.

# [8 Промышленная экология](#_Toc474749010)

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 8.1.

Таблица 8.1 – 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) | 0,14 м3/день |
| Количество (объем) образования твердых бытовых отходов, т (м3) /день | 0,08 м3/ 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 |

В таблице 8.2 приведены экологические аспекты деятельности и виды воздействия экологических аспектов на окружающую среду, а также возможные мероприятия по сокращению воздействия.

Таблица 8.2 – Экологические аспекты деятельности и виды воздействия экологических аспектов на окружающую среду

|  |  |  |
| --- | --- | --- |
| Экологический аспект | Воздействие на окружающую среду (работающих) | Предложения по сокращению воздействия |
| Освещенность | Расход энергии | Рациональное использование электроэнергии |
| Отработанные лампы | Загрязнение тяжелыми 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 | Твердые отходы производства | Раздельный сбор. Переработка вторичного сырья |
| Сточная вода (бытовая) | Загрязнение гидросферы | Установка счетчика, фильтра, использование рециркуляции бытовой воды |

Схема материальных потоков при работе с веб-сервисом представлена на рисунке 8.1.

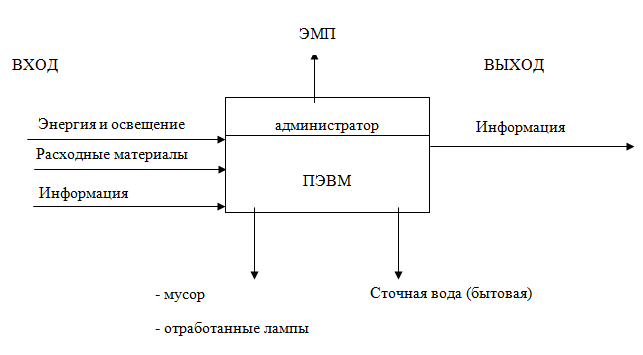


Рисунок 8.1 – Схема материальных потоков при работе с веб-сервисом

Утилизация компьютерной и офисной техники – это передовой подход к сохранению окружающей среды с пользой для бюджета. Речь идет о специальной услуге, которую оказывают профильные компании, принимая устаревшие единицы электрооборудования и комплектующие для дальнейшей переработки. Сдача компьютерной техники на переработку целесообразна и в рамках небольших компаний, и в крупных международных корпорациях.

Утилизация устаревшей компьютерной техники включает несколько этапов:

* оценка технического состояния устройств и составление "дефектных" актов;
* перевозка утилизируемой техники;
* извлечение ценных деталей и материалов из списанных устройств;
* сортировка;
* упаковка;
* переработка полученного лома с последующим извлечением драгоценных металлов.
* Пример. На переработку поступает 20 кг печатных плат персональных компьютеров поколения Pentium (материнские платы). Во вращающийся барабан объемом 150 л, оборудованный электрическим подогревом, загружают 20 кг печатных плат с радиодеталями навесного монтажа, помещенные в контейнер из металлической сетки с ячейкой 5×5 мм, и приливают 40 л щелочного водного раствора, содержащего 40% NaOH и 10% KOH.

Процесс растворения лака с поверхности плат ведут при перемешивании и температуре 70°C в течение 2 ч. Затем для хлопьеобразования и коагуляции лакового покрытия добавляют катионный флокулянт «Праестол 650» из расчета 0,2 кг/м3и дополнительно перемешивают в течение 1 часа. Полученный щелочной раствор сливают и отстаивают. После чего осветленную часть раствора вновь используют для обработки новых порций печатных плат, а сгущенную часть накапливают и утилизируют.

В результате достигается полное растворение лакового покрытия с печатных плат. Обработанные таким образом платы промывают в барабане водой и для растворения оловянного припоя заливают 10%-ный раствор метансульфоновой кислоты и процесс ведут по способу-прототипу.

Полученную суспензию метаоловянной кислоты в растворе метансульфоновой кислоты коагулируют путем введения добавки ПАВ с последующим кипячением в течение 30 мин. После охлаждения раствор декантируют от осевшей метаоловянной кислоты, а платы и взвесь метаоловянной кислоты сепарируют.

Выделенную таким образом метаоловянную кислоту отфильтровывают на вакуумном фильтре, промывают водой, сушат и прокаливают при температуре 800°C с получением товарного продукта оксида олова, а из раствора метансульфоновой кислоты осаждают сульфат свинца серной кислотой.

Полученный фильтрат корректируют по содержанию метансульфоновой кислоты и повторно используют для растворения припоя следующих порций плат. После сепарации платы вновь промывают водой и сушат.

Затем платы загружают во вращающийся барабан и приливают 30 л раствора, содержащего 30% NaCl и 20% CuCl2. Растворение меди с поверхности плат ведут при слабом перемешивании в течение 60 мин и температуре 80°C. Обработанные таким образом платы промывают водой и далее используют по назначению, а растворы отправляют на электрохимическую переработку в электролизер.

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

# 9 Ресурсосбережение

Ресурсосбережение – система научно-технических, организационных, экономических и воспитательных мер, направленных на наиболее рациональное и эффективное использование всех видов ресурсов.

Основным ресурсом, который потребляет компьютерная техника, является электроэнергия. Подавляющее большинство технических средств механизации и автоматизации производственных процессов (оборудование, приборы ЭВМ), замена человеческого труда машинным в быту имеют электрическую основу.

Основными принципами государственной политики Республики Беларусь в сфере ресурсосбережения являются:

- осуществление государственного надзора за рациональным использованием топливно-энергетических ресурсов;

- разработка государственных и межгосударственных научно-технических, республиканских, отраслевых и региональных программ энергосбережения и их финансирование;

- создание системы финансово-экономических механизмов, обеспечивающих 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 государственной экспертизы энергетической эффективности проектных решений;

- создание и широкое распространение экологически чистых и 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.

Основные направления энергосбережения регламентируются международными, межгосударственными и государственными нормативными правовыми актами, техническими нормативными правовыми актами, утвержденными специально уполномоченными государственными органами надзора и контроля.

Для реализации ресурсосбережения при работе с компьютером необходимо экономно использовать электроэнергию. Компьютеры потребляют много энергии, тем более, зачастую от них требуется оставаться включенными сутками. Наибольшее количество энергии тратится на поддержание работы монитора и жесткого диска. Поэтому в любой операционной системе есть несколько энергосберегающих режимов, использование которых может значительно сократить потребляемую энергию и, как следствие, ресурсы и деньги.

Например, в операционной системе «Windows», начиная с версии «Windows 7», существуют три энергосберегающих режима – сон, гибернация (спящий режим) и гибридный сон.

Сон – это энергосберегающий режим, позволяющий компьютеру за несколько секунд вернуться во включенное состояние. При переходе в режим сна, питание компьютера не отключается полностью, а лишь переходит на сниженное энергопотребление. Открытые программы и документы сохраняются в оперативной памяти, чтобы сразу после вывода компьютера из режима сна пользователь смог возобновить работу. Если во время сна питание компьютера будет полностью отключено, то все несохраненные настройки и изменения файлов будут утрачены.

Гибернация (спящий режим) – это энергосберегающий режим, разработанный специально для ноутбуков. В отличие от режима сна, помещающего открытые программы и документы в оперативную память, спящий режим сохраняет открытые документы и программы на жесткий диск (в файл hiberfil.sys) и затем переводит компьютер в режим сниженного энергопотребления. Все настройки и изменения в документах после выхода из режима гибернации сохранятся даже при полном отключении питания. Выход компьютера из спящего режима происходит обычно быстро – быстрее, чем включение «Windows» после завершения работы, но дольше, чем выход из режима сна. Все открытые на момент входа в спящий режим документы и программы восстанавливаются из файла «hiberfil.sys», после чего вы сразу можете вернуться к работе, продолжив её с того места, где вы остановились.

Гибридный сон разработан специально для настольных компьютеров. Режим гибридного сна представляет собой комбинацию режимов сна и гибернации – режим гибридного сна помещает ваши настройки, открытые документы и программы в оперативную память и на жесткий диск, после чего компьютер переходит в режим пониженного энергопотребления. Вы сможете быстро вывести компьютер из состояния гибридного сна и продолжить работу. Все настройки и изменения в документах после выхода из режима гибридного сна сохранятся даже при полном отключении питания. Обычно режим гибридного сна на настольных компьютерах по умолчанию включен.

Все современные операционные системы поддерживают механизмы энергосбережения, позволяющие выключать отдельные компоненты после определённого периода бездействия. Жёсткие диски могут останавливать вращающиеся пластины, мониторы могут выключаться, да и весь компьютер может переходить в режим ожидания или даже гибернации. Последний способ является весьма эффективным для выключения системы, поскольку при повторном включении содержимое памяти до гибернации считывается с жёсткого диска, поэтому заново операционная система не загружается.

Наконец, пользователь тоже может немало сделать, начиная от включения отдельных устройств только тогда, когда это нужно, и заканчивая характером своей деятельности в рабочем или игровом окружении. Компьютер, который ничего не делает, лучше перевести в состояние гибернации или выключить, а внешнюю периферию лучше выключать или переводить в режим ожидания, когда она не нужна.

# Заключение

В результате выполнения данного дипломного проекта было разработано веб-приложение для управления записями на медицинские услуги. Приложение позволяет пользователям просматривать список доступных услуг, записываться на прием к врачам, а также отслеживать свои записи и историю посещений. Веб-приложение также предоставляет функции авторизации, регистрации и управления аккаунтом пользователя.

Данное веб-приложение обладает интуитивно понятным и функциональным интерфейсом, что обеспечивает удобство использования для пользователей с разными уровнями опыта. Разработанное приложение соответствует всем предъявленным требованиям и может быть использовано в качестве платформы для упрощения процесса записи на медицинские услуги и координации работы медицинских учреждений.

Приложение является веб-приложением, что позволяет обеспечить одновременный доступ и взаимодействие с сервером для множества пользователей, расположенных в разных местах. Вся используемая информация хранится в базе данных, специально разработанной для данного проекта. Доступ к базе данных осуществляется только через веб-сервис, что гарантирует безопасность данных.

Безопасность данных также обеспечивается путем разграничения прав доступа и системы авторизации. Различные роли пользователей, такие как администратор, медицинский персонал и пациенты, имеют разные уровни доступа к функциональности приложения.

Разработанное приложение позволяет производить учет медицинских услуг, специалистов и записей на прием, а также облегчает взаимодействие между пациентами и медицинскими учреждениями. Это содействует повышению качества и оперативности предоставления медицинских услуг.

В процессе разработки данного дипломного проекта были использованы современные технологии и фреймворки, что обеспечивает быстроту и гибкость при разработке, а также возможность дальнейшего расширения и модернизации приложения.

# Список использованной литературы

1. Больницы [Электронный ресурс]. – Режим доступа: https://minzdrav.gov.by/ru/dlya-belorusskikh-grazhdan/uchrejdenia-zdravoohranenia/bolnitsy.php. – Дата доступа: 07.05.2023.
2. Модели функционирования [Электронный ресурс]. – Режим доступа: http://econtool.com/modeli-funktsionirovaniya.html. – Дата доступа: 07.05.2023.
3. Жизненный цикл программного продукта [Электронный ресурс]. – Режим доступа: https://it-black.ru/zhiznennyy-tsikl-programmnogo-produkta/. – Дата доступа: 07.05.2023.
4. DocDoc [Электронный ресурс]. – Режим доступа: https://www.docdoc.ru/. – Дата доступа: 07.05.2023.
5. Zocdoc [Электронный ресурс]. – Режим доступа: https://www.zocdoc.com/. – Дата доступа: 07.05.2023.
6. Medesk [Электронный ресурс]. – Режим доступа: https://www.medesk.ru/. – Дата доступа: 07.05.2023.
7. Bookimed [Электронный ресурс]. – Режим доступа: https://bookimed.com/. – Дата доступа: 07.05.2023.
8. Healthgorilla [Электронный ресурс]. – Режим доступа: https://www.healthgorilla.com/. – Дата доступа: 07.05.2023.
9. Серверные языки: Java (обзор) [Электронный ресурс]. – Режим доступа: https://codomaza.com/article/servernye-jazyki-java-obzor. – Дата доступа: 07.05.2023.
10. Spring Framework [Электронный ресурс]. – Режим доступа: https://javastudy.ru/frameworks/spring/. – Дата доступа: 07.05.2023.
11. HTML [Электронный ресурс]. – Режим доступа: https://loftschool.com/blog/posts/html-chto-eto. – Дата доступа: 07.05.2023.
12. Основы CSS [Электронный ресурс]. – Режим доступа: https://college.arthur-nesterenko.dev/css/basic. – Дата доступа: 07.05.2023.
13. UML [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/4533052/. – Дата доступа: 07.05.2023.
14. Клиентская часть веб приложения [Электронный ресурс]. – Режим доступа: https://pastebin.com/t0K3YZNK. – Дата доступа: 07.05.2023.
15. Spring — MVC Framework [Электронный ресурс]. – Режим доступа: https://krishnabankar.medium.com/spring-mvc-framework-e611b6d91048. – Дата доступа: 07.05.2023.
16. Нормализация отношений [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/4165581/page:2/. – Дата доступа: 07.05.2023.
17. Диаграмма вариантов использования как концептуальное представление бизнес-системы в процессе ее разработки. [Электронный ресурс]. – Режим доступа: https://intuit.ru/studies/courses/32/32/lecture/1004. – Дата доступа: 07.05.2023.
18. Что такое системное тестирование [Электронный ресурс]. – Режим доступа: https://deepcloud.ru/articles/chto-takoe-sistemnoe-testirovanie/. – Дата доступа: 07.05.2023.