Министерство образования Республики Беларусь

ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «ИСАП»

ОТЧЁТ

по преддипломной практике

|  |
| --- |
| Выполнил: |
| студент группы Ит-6 |
| Лапко М. Л. |
| Руководитель от университета: |
| Дунина Е. Б. |
| Руководитель от предприятия ООО «Фабрика инноваций и решений»: |
| Медюха В. В. |

**Витебск 2022**

**Оглавление**

[Введение 3](#_Toc7648)

[1 Анализ объекта 4](#_Toc17039)

# Введение

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

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

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

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

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

Сформулируем основные задачи преддипломной практики:

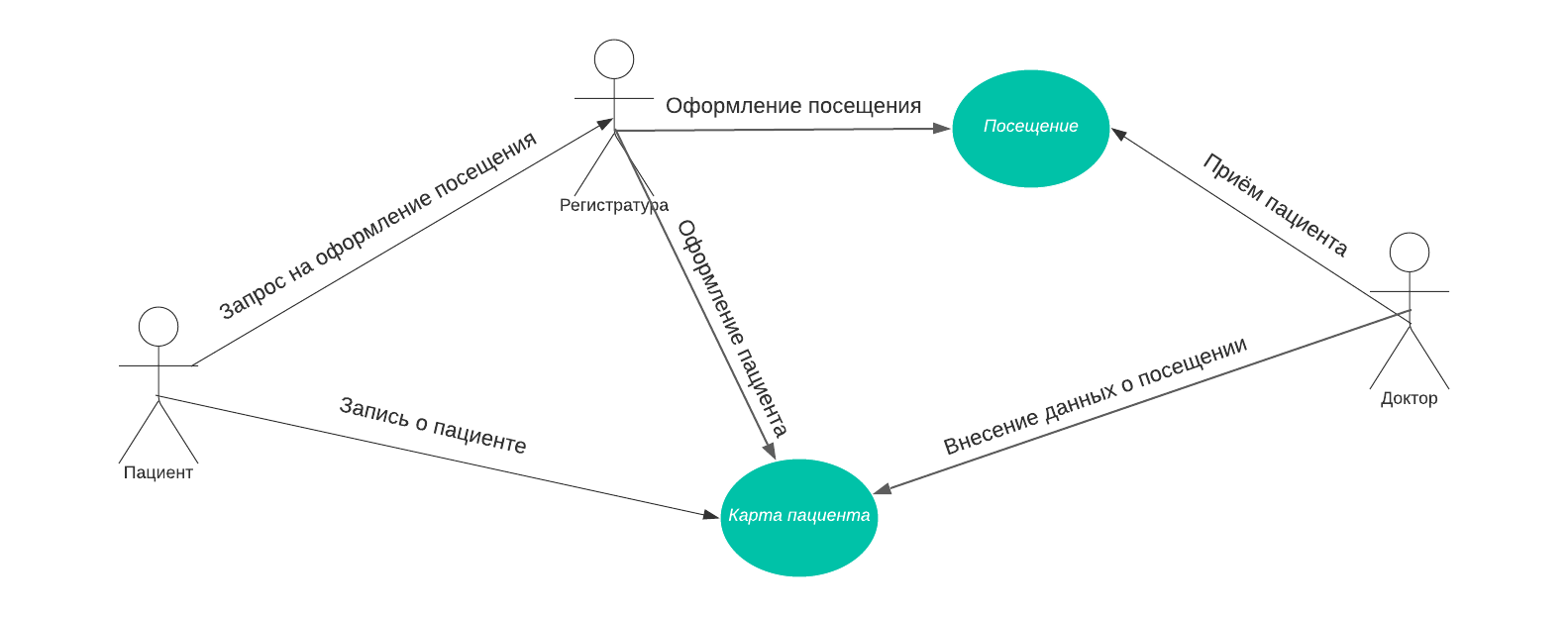
* провести анализ предметной области;
* проанализировать и выбрать инструменты разработки программной системы;
* выполнить проектирование системы;

# 1 Анализ объекта

## Описание предметной области

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

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

Рисунок 1.1 - Диаграмма прецедентов

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

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

## Построение концептуальной модели предметной области

Концептуальная модель разрабатываемой системы представлена ниже (рисунок 1.2).

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

После авторизации/регистрации пользователь перенаправляется на главную страницу, которая отличается для доктора и пациента. Доктор может посетить свой профиль для редактирования информации, просмотреть уведомления, которые приходят при оформлении нового посещения к данному специалисту. Также доктор может просмотреть предстоящие для него посещения, каждое посещение содержит в себе ссылку на карту пациента, которую доктор также может просмотреть для ознакомления с пациентом и его заболеваниями. Доктор может редактировать информацию о посещении, например, указывать заболевание, которое было найдено у пациента в результате приёма или добавлять определённое описание на своё усмотрение.

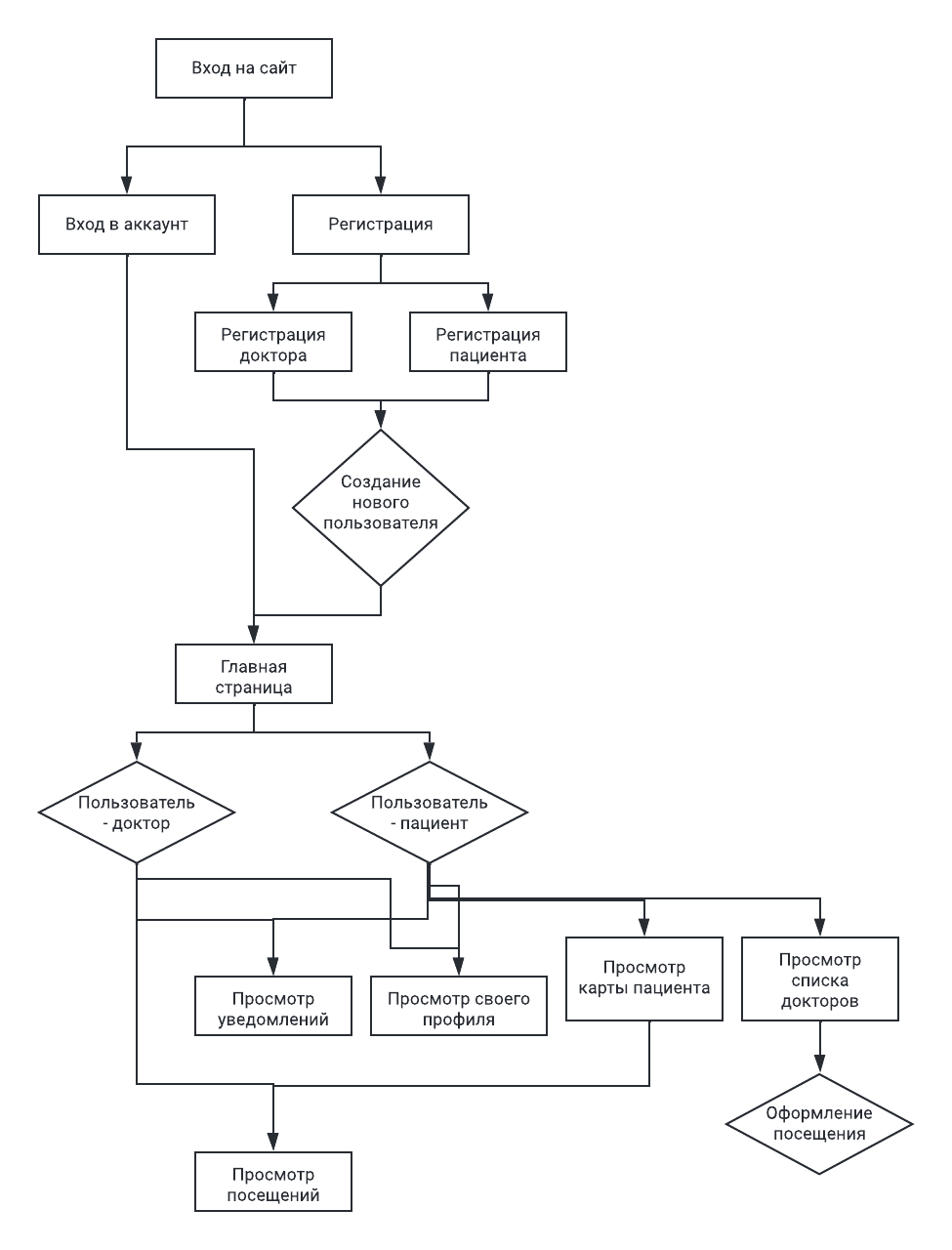


Рисунок 1.2 - Концептуальная модель

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

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

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

# 2 Постановка задачи

## 2.1 Определение требований к программной системе

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

Возможности доктора:

- просмотр своего профиля;

- возможность редактирования профиля;

- возможность удаления профиля;

- просмотр уведомлений;

- просмотр предстоящих посещений;

- возможность редактирования самого посещения.

Возможности пациента:

- просмотр своего профиля;

- возможность редактирования профиля;

- возможность удаления профиля;

- просмотр уведомлений;

- просмотр списка докторов;

- возможность оформить запрос на посещение специалиста;

- просмотр своей карты;

- просмотр своих посещений;

Дополнительные требования:

- возможность новому пользователю зарегистрироваться как доктор или пациент;

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

## 2.2 Описание аналогов системы

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

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

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

- omskzdrav.ru - региональный портал медицинских услуг, содержит возможность записи на приём к специалисту.

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

Также данные интернет ресурсы в большинстве своём имеют дополнительные возможности, зачастую относящиеся к определённой поликлинике, например: платные услуги, вызов врача на дом и так далее.

## 2.3 Обзор и обоснование выбора инструментальных средств

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

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

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

Вот лишь несколько функций языка C#, которые позволяют создавать надёжные и устойчивые приложения. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения поддерживают приёмы функционального программирования. Синтаксис LINQ создаёт общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В C# имеется Единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощённые структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода [1].

Язык C# практически универсален. Можно использовать его для создания любого ПО: продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android [2].

Также в языке присутствует обилие синтаксического сахара, который делает тяжёлую жизнь программиста капельку слаще. Вместо того, чтобы писать 100500 строк кода, присутствует возможность использовать готовую конструкцию, а компилятор сделает всю остальную работу. Но некоторые такие конструкции являются не самыми оптимальными с точки зрения производительности. Но все это перекрывается за счёт удобочитаемости кода и высокой скоростью разработки [3].

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

ASP.NET Core — это веб-инфраструктура с открытым исходным кодом, оптимизированная для облачных вычислений, для разработки современных веб-приложений, которые можно разрабатывать и запускать на Windows, Linux и Mac. Он включает в себя инфраструктуру MVC, которая теперь объединяет функции MVC и веб-API в единую среду веб-программирования. Он был переработан с нуля, чтобы быть быстрым, гибким, современным и работать на разных платформах. В дальнейшем ASP.NET Core — это фреймворк, который можно использовать для веб-разработки с .NET. Если у вас есть опыт работы с MVC или веб-API за последние несколько лет, вы заметите некоторые знакомые функции [4].

Бэкэнд часть является веб-API. Платформа веб-API ASP.NET позволяет с легкостью создавать службы HTTP для широкого диапазона клиентов, включая браузеры и мобильные устройства. ASP.NET Web API - это идеальная платформа для сборки REST-приложений на базе .NET Framework [5].

Для написания приложения была выбрана среда программирования Microsoft Visual Studio.

Microsoft Visual Studio является средой программирования, разработанной компанией Microsoft. Эта среда позволяет создавать кроссплатформенные проекты на различных языках программирования, таких как Visual Basic, Visual C#, Visual C++, Visual F# и другие. Также она позволяет создавать программы, использующие в своей работе платформу .NET, которая позволяет использовать большой набор сервисов, реализующихся в виде промежуточного, независящего от базовой архитектуры, кода. Основной целью создания платформы .NET является возможность реализации разработчиками специальных сервисно-ориентированных программ, работающих на любых платформах.

MS Visual Studio позволяет разработчику иметь доступ к огромной коллекции различных функций, которые позволяют вести разработки для любой версии операционной системы семейства Windows, для интернет-приложений и мобильных приложений. Также среда программирования открывает широкие возможности в области облачных технологий. Эта среда открывает разработчику широкие возможности для реализации самых разных проектов, реализуя высокую производительность и независимость от особенностей оборудования [6].

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

React — библиотека, написанная на JavaScript, которая используется для работы с интерфейсами. В 2011 году её начали использовать для социальной сети Facebook, а уже в 2013 году библиотеку выложили в открытый доступ и энтузиасты со всего мира начали создавать инструменты для расширения её возможностей. Разработчики используют React, чтобы создавать интерфейсы, которые способны менять контент без перезагрузки страницы. Благодаря этому сайты или нативные приложения быстро отзываются на действия пользователей. Можно добавлять товары в корзину без перезагрузки страницы или заполнять формы без переадресации [7].

React предпочтительнее использовать по следующим причинам:

* Технология SPA (single-page application, по-русски: “разработка одностраничных приложений”): React поможет вам создать одностраничное приложение. С помощью ReactJS вы сможете изменять (управлять/манипулировать) контент всей страницы с минимальным кодом.
* Декларативный подход: React использует декларативный дизайн со всем синтаксическим сахаром, что помогает написать поддерживаемый код высокого уровня. Вам просто нужно определить цель, и React будет обрабатывать инструкции JavaScript DOM с учетом ситуаций, в которых они используются.
* Компонент-управляемый пользовательский интерфейс: React основан на компонентной концепции. Компоненты являются многоразовыми строительными блоками в пользовательском интерфейсе. С помощью ReactJS вы можете создать инкапсулированный компонент, который управляет своими данными, и избежать сценария влияния на другие состояния и действия компонентов в DOM-дереве. Это только одна из характеристик компонент-управляемого интерфейса. Он также помогает повторно использовать код, разделять ответственность и избегать повторения [8].

TypeScript (TS, TScript или «тайпскрипт») — это язык программирования для веб-разработки, основанный на JavaScript. Делает код понятнее и надежнее, добавляет статическую типизацию (переменные привязаны к конкретным типам данных), а также может быть скомпилирован в JavaScript. TypeScript используют фронтенд- и бэкенд-разработчики.

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

TypeScript помогает сократить время на выявление и устранение багов, которые иногда сложно найти в динамической среде JavaScript. С помощью TypeScript можно написать более понятный и читаемый код, который максимально описывает предметную область. Таким образом архитектура становится более выраженной [9].

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

Visual Studio Code - это «бесплатный редактор, который помогает программисту писать код, помогает в отладке и исправлении кода с помощью метода intelli-sense». В обычных условиях это облегчает пользователю написание кода простым способом. Многие говорят, что это половина IDE и редактора, но решение остается за программистами. Visual Studio Code поддерживает несколько языков программирования, имеет кроссплатформенную поддержку, огромное количество расширений для упрощения создания кода, а также встроенную поддержку системы контроля версий Git [10].

# 3 Проектирование

## 3.1 Разработка архитектуры программного продукта