**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01

«Информатика и технологии программирования»

**Отчет по преддипломной практике**

на тему:  
«Программный комплекс по выбору лекарственных препаратов на основе экс-пертных знаний»

Исполнитель: студент гр. ИП-42

Пинчук В.А.

Руководитель от предприятия:

Гуд С.Н.

Руководитель: ст. преподаватель

Косинов Г.П.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2020

# **СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc38123146)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ ПО ВЫБОРУ ЛЕКАРСТВЕННЫХ ПРЕПАРАТОВ 4](#_Toc38123147)

[1.1 Анализ предметной области 4](#_Toc38123148)

[1.2 Обзор аналогов 5](#_Toc38123149)

[1.3 Анализ инструментария реализации задачи 10](#_Toc38123150)

[1.4 Требования к проектируемому программному обеспечению 12](#_Toc38123151)

[2 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 13](#_Toc38123152)

[2.1 Функциональная модель программного комплекса 13](#_Toc38123153)

[2.2 Архитектура системы 17](#_Toc38123153)

[2.3 Информационная модель 18](#_Toc38123157)

[ЗАКЛЮЧЕНИЕ 24](#_Toc38123158)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc38123159)

Введение

Здоровье – это самое важное составляющее, каждого человека. Сегодня, когда у человека на счету каждая минута, часто люди не обращают внимание на состояние своего здоровья, что со временем может привести к непоправимым последствиям. Также многие люди, при возникновении каких-либо проблем со здоровьем, не знают что предпринять и вынуждены обратиться к специалисту, даже если это элементарная простуда.

Данное приложение предоставляет экспертную помощь при различных заболеваниях. Приложение помогает определить являются ли симптомы опасными, для вашего здоровья, требующие профессиональную медицинскую помощь или вы можете справиться с этим самостоятельно.

Вы чувствуете недомогание? У вас тяжесть в желудке? Вы съели ягоду и почувствовали тошноту? У вас повышенная температура? Не нужно гадать, что вам нужно делать. Получите точные ответы, подготовленные экспертами в медицине. Используйте данное приложение, чтобы узнать, нужно ли вам вызвать неотложную скорую медицинскую помощь, нужно ли вам обратиться в ближайшее учреждение здравоохранения или можно просто остаться дома, следуя определенным правилам. Поиск заболеваний происходит на основе общих введенных данных пользователя, такие как возраст, вес, пол, а также симптомы.

Также, если вы принимаете много различных лекарственных средств и сомневаетесь в эффективности препаратов, при совместном их использовании, или у вас появились побочные эффекты, вы должны проверить все препараты на совместимость. Для проверки лекарств на совместимость вы должны прочитать огромное количество инструкций к каждому препарату, что займет не мало времени. Данное приложение позволит проверить лекарственные средства на совместимость намного быстрее и более информативно.

Целью данного дипломного проекта является создание такой автоматизированной системы, которая бы смогла:

* предоставить возможность создания актуальной базы заболеваний и методы их лечения, доступной любому пользователю;
* экономить время пользователей, упрощая процесс назначения лечения;
* частично разгрузить различные органы здравоохранения;
* сделать проверку лекарств на совместимость проще и быстрее.

1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ ПО ВЫБОРУ ЛЕКАРСТВЕННЫХ ПРЕПАРАТОВ

* 1. Анализ предметной области

Лекарственное средство — вещество или смесь веществ [синтетического](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B5%D1%82%D0%B8%D0%BA%D0%B0) или природного происхождения в виде [лекарственной формы](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0) ([таблетки](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B5%D1%82%D0%BA%D0%B8), [капсулы](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D0%B0_(%D0%BB%D0%B5%D0%BA%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)), [раствора](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%82%D0%B2%D0%BE%D1%80_(%D0%BB%D0%B5%D0%BA%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)), [мази](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B7%D0%B8) и т. п.), применяемые для [профилактики](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_(%D0%BC%D0%B5%D0%B4%D0%B8%D1%86%D0%B8%D0%BD%D0%B0)), диагностики и [лечения](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [заболеваний](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B1%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Перед употреблением в медицинской практике лекарственные средства должны проходить [клинические исследования](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и получать разрешение к применению.

Знание механизма действия позволяет осмысленно выбрать необходимый препарат для лечения конкретного заболевания. Кроме того, знание механизма действия необходимо также для правильного комбинирования лекарств и предвидения возможного возникновения нежелательных эффектов.

Знание фармакокинетики лекарственного средства дает врачу возможность осуществить индивидуальный подбор лекарственной терапии данному больному, исходя из особенностей функционирования его организма. Кроме того, знание фармакокинетики препарата позволяет предвидеть появление нежелательных эффектов, а также помогает выбрать оптимальный режим дозирования при данном пути введения для того, чтобы обеспечить терапевтическую концентрацию лекарственного вещества в области рецептора.

Активным средством в руках врача, с помощью которого он может влиять на течение того или иного заболевания, является лекарственный препарат.

Лекарственное средство в руках знающего врача приносит огромную пользу людям. Незнание лекарственных средств, неумение пользоваться ими, низкие морально-этические требования к себе могут привести к непоправимым последствиям для больного человека.

Нередко перед врачом стоит сложная задача – выбрать из большого арсенала лекарственных средств не только самое эффективное, но и наименее токсичное, а также уменьшить риск появления побочного действия. Это в значительной мере обусловлено тем, что при различных условиях одно и то же вещество может оказаться лекарством или ядом. Так, стрихнин, морфин и другие ядовитые и сильнодействующие лекарственные вещества в сравнительно небольших, так называемых терапевтических дозах оказывают лечебный эффект. С увеличением доз этих лекарственных средств выше допустимых они могут проявлять токсическое действие, нередко приводящее к тяжелым последствиям. Иногда обычные дозы лекарственных веществ вместо желаемого действия могут оказать отрицательное влияние на организм, что связывают с индивидуальной чувствительностью больных к этому лекарственному веществу.

Любое лекарственное средство, покупаемое в аптеке, сопровождено специ­альной инструкцией по применению. Но часто ли мы внимательно относимся к этой информации? Между тем, соблюдение или несоблюдение правил приема может оказать большое, если не решающее, влияние на действие лекарства. Например, при приеме внутрь пища, а также желудочный сок, пищеварительные ферменты, желчь, которые выделяются в процессе ее переваривания, могут взаимодействовать с лекарственными веществами и изменять их свойства. Именно поэтому совсем не безразлично, когда лекарство будет принято: натощак, во время или после еды.

* 1. Обзор аналогов

На данный момент существует несколько web-приложений, которые упрощают подбор и поиск лекарственных средств. Данные приложения позволяют быстро найти подробную информацию о нужном препарате, а также найти средства их реализаций. В качестве аналога, были рассмотрены следующие web-приложения:

* https:// tabletka.by/ - приложение по поиску лекарственных средств в Беларуси
* https://103.by/ - международная медицинская платформа

***1.2.1*** Приложение tabletka.by предоставляет услуги, позволяющие пользователю производить поиск лекарственных средств и товаров аптечного ассортимента в данных, полученных от аптек в рамках соглашения между аптеками и приложением tabletka.by.

A screenshot of a computer

Description automatically generated

Рисунок 1.1 – Главная страница приложения tabletka.by

Для удобства пользования сайтом данные по лекарственным средствам, медицинской технике, товарам аптечного ассортимента систематизируются и приводятся к единому написанию. Прочий ассортимент представлен в поиске в написании справочников аптек.

A screenshot of a cell phone

Description automatically generated

Рисунок 1.2 – Данные по лекарственному средству

Также приложение предоставляет данные о аптеках, где можно приобрести данный препарат. Для каждой аптеки указан адрес, телефон, график работы и цена препарата. Есть возможность забронировать препарат на определенный промежуток времени. Также есть возможность просмотреть местоположение аптеки на карте.

A screenshot of a cell phone

Description automatically generated

Рисунок 1.3 – Список аптек реализующие продажу необходимого препарата

A screenshot of a computer

Description automatically generated

Рисунок 1.4 – Местоположение аптеки на карте

Любая аптека может подключиться к данному сервису. Для подключения к сервису требуется наличие программы предметно-количественного учета с возможностью выгрузки отчета по остаткам и подключение к сети интернет. Подробная информация о формате выгрузки и реквизитам подключения определяется разработчиками сервиса.

***1.2.2*** Международная медицинская платформа 103 — проект компании ARTOX, специализирующейся на тематических агрегаторах. ARTOX осуществляет свою деятельность с 2007 года, а с 2012 года является резидентом Парка высоких технологий в Беларуси.

В 2013 году появился 103.BY — белорусский портал и мобильное приложение с максимально полным каталогом медицинских учреждений, сервисом поиска лекарств в аптеках и полезным контентом о здоровье и красоте.

A screenshot of a social media post

Description automatically generated

Рисунок 1.5 – Главная страница приложения 103

К сервису поиска лекарств подключены более 3 600 аптек по всей Беларуси, и их количество постоянно растет. Любой пользователь легко может найти нужное лекарство по минимальной цене в выбранном регионе.

A screenshot of a cell phone

Description automatically generated

Рисунок 1.6 – Список найденных лекарственных средств

Также аптеки могут сортироваться по близости к текущему местонахождению пользователя и отображаться на карте. Доступны функции онлайн-бронирования, поиска наборов лекарств (можно выбрать сразу несколько препаратов и найти, где полный набор есть в наличии по минимальной цене), инструкции к препаратам, аналоги лекарств.

A close up of a map

Description automatically generated

Рисунок 1.7 – Список аптек реализующие необходимый препарат

Каждое учреждение имеет персональную страницу на отдельном поддомене, которая фактически является сайтом учреждения и содержит всю необходимую для пользователей информацию.

Так, по каждому медицинскому учреждению будет как минимум справочная информация: адрес, контакты, время работы, отзывы. Для расширенного формата доступен полный функционал: сведения об организации, перечень услуг, их описание и цены, фото- и видеогалереи, данные о специалистах, новости, акции и скидки.

* 1. Анализ инструментария реализации задачи

Весь проект можно выделить четыре основные части, а именно:

* создание браузерного приложения;
* создания Rest Api приложения, для отправки и обработки данных, а также для взаимодействия с базой данных;
* создание сервиса, для интеграции со сторонними сервисами, предоставляющие базу лекарственных средств;
* проектирование и создание базы данных.

Для создания браузерного приложения были применены следующие технологии: React, Redux, SCSS, Typescript, Webpack, Lodash, Babel.

Для создания Rest Api приложения и сервиса для интеграции были применены технологии: Node.js, Typescript, Express, Sequelize.

Для хранения данных была применена база данных MySql. Для удобного управления базой данных было использовано специальное средство администрирования Workbench.

При создании проекта, использовалась среда разработки JetBrains WebStorm.

React - это библиотека JavaScript, которая используется для создания пользовательского интерфейса. React был создан компанией Facebook. Данная библиотека позволяет создавать одностраничные приложения. Также особенностью данной библиотеки является, использование Virtual Document Object Model. Он создает кеш структуру страницы в памяти, с ее помощью находит разницу между прошлым и текущем состоянием интерфейса для оптимизации обновления Document Object Model. Таким образом страница обновляется не полностью, а только некоторая её часть, за счет чего увеличивается скорость работы приложения.

Сегодняшний мир веб-сайтов трудно представить без языка JavaScript. Это то, что делает живыми веб-страницы, которые мы каждый день просматриваем в своем веб-браузере. Первоначально JavaScript обладал довольно небольшими возможностями. Его цель состояла лишь в том, чтобы добавить немного поведения на веб-страницу. Например, обработать нажатие кнопок на веб-странице, произвести какие-нибудь другие действия, связанные прежде всего с элементами управления.

Однако развитие веб-среды, появление HTML5 и технологии Node.js открыло перед JavaScript гораздо большие горизонты. Сейчас JavaScript продолжает использоваться для создания веб-сайтов, только теперь он предоставляет гораздо больше возможностей. Также он применяется как язык серверной стороны. JavaScript является языком с динамической типизацией, что дает определенные плюсы и минусы. В больших проектах отсутствие статической типизации ведет к большему количеству багов. Для решения данной проблемы была разработана надстройка TypeScript.

TypeScript - это строго типизированный и компилируемый язык представленный на основе JavaScript. Хотя на выходе компилятор создает все тот же JavaScript, который затем исполняется браузером. Однако строгая типизация уменьшает количество потенциальных ошибок, которые могли бы возникнуть при разработке на JavaScript.  TypeScript реализует многие концепции, которые свойственны объектно-ориентированным языкам, как, например, наследование, полиморфизм, инкапсуляция и модификаторы доступа и так далее.

Часто React приложение используется с Redux. Redux — библиотека для JavaScript с открытым исходным кодом, предназначенная для управления состоянием приложения. Данная библиотека реализует архитектуру единого хранилища данных в приложении.

Препроцессор SCSS позволяет использовать функции недоступные в самом CSS, например, переменные, вложенности, миксины, наследование и другие приятные вещи.

Node.js представляет среду выполнения кода на JavaScript, которая построена на основе движка JavaScript Chrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript.

Библиотека Express значительно упрощает создание Rest Api приложения на Node.js. Она упрощает работу с роутингом, отправкой статических файлов, а также упрощает работу с запросами.

ORM – это технология программирования, который связывает базу данных с моделью объектно-ориентированных языков программирования. Он создает для СУБД виртуальный объектный интерфейс. Библиотека Sequelize – это Node.js ORM система для работы с такими базами данных, как PostgreSQL, MySQL, SQLite и MSSQL.

  Webpack представляет популярный упаковщик модулей, который позволяет автоматизировать создание из нескольких файлов один.

Babel – транспайлер, переписывающий код на ES-NEXT в код на предыдущем стандарте.

* 1. Требования к проектируемому программному обеспечению

На основании состояния предметной области, анализа используемых в ней методах и инструментах, было сформированы требования к разрабатываемой системе

Цель разработки: разработать кроссплатформенное веб-приложение по выбору лекарственных препаратов на основе экспертных знаний.

Требуемый функционал:

* регистрация, авторизация пользователей;
* система ролей;
* создание заболевания, с необходимым описанием, добавлением лечения и добавлением рекомендаций;
* поиск заболеваний по определенным фильтрам;
* привязка несовместимых лекарств;
* проверка лекарств на совместимость.

В приложении присутствует система ролей. Для всех ролей создано единое браузерное приложение с применением технологии Lazy Loading, что позволит не создавать для каждой роли отдельное приложение и уменьшить время загрузки приложения.

Приложение должно иметь UI/UX дизайн для обычного пользователя. Приложение должно работать во всех современных браузерах.

Список используемых технологий и их обоснование представлены в разделе 1.3 (Анализ инструментария реализации задачи). Таким образом, в данном разделе выполнен анализ предметной области, подготовлено общее техническое задание, определены технологии разработки приложения и обоснован их выбор.

1. Архитектура программного обеспечения
   1. Функциональная модель программного комплекса

В техническом задании было указано, что программный комплекс будет разрабатываться как клиент-серверное приложение.

Клиент-сервер – это, сетевая архитектура, в которой сетевая нагрузка распределена между сервером и клиентом. Обычно они расположены на разных вычислительных машинах и взаимодействуют между собой посредством сетевых протоколов, но также могут быть расположены и на одной вычислительной машине.

Преимущества такой системы:

* отсутствие дублирования кода программы-сервера программами-клиентами;
* так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются;
* все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

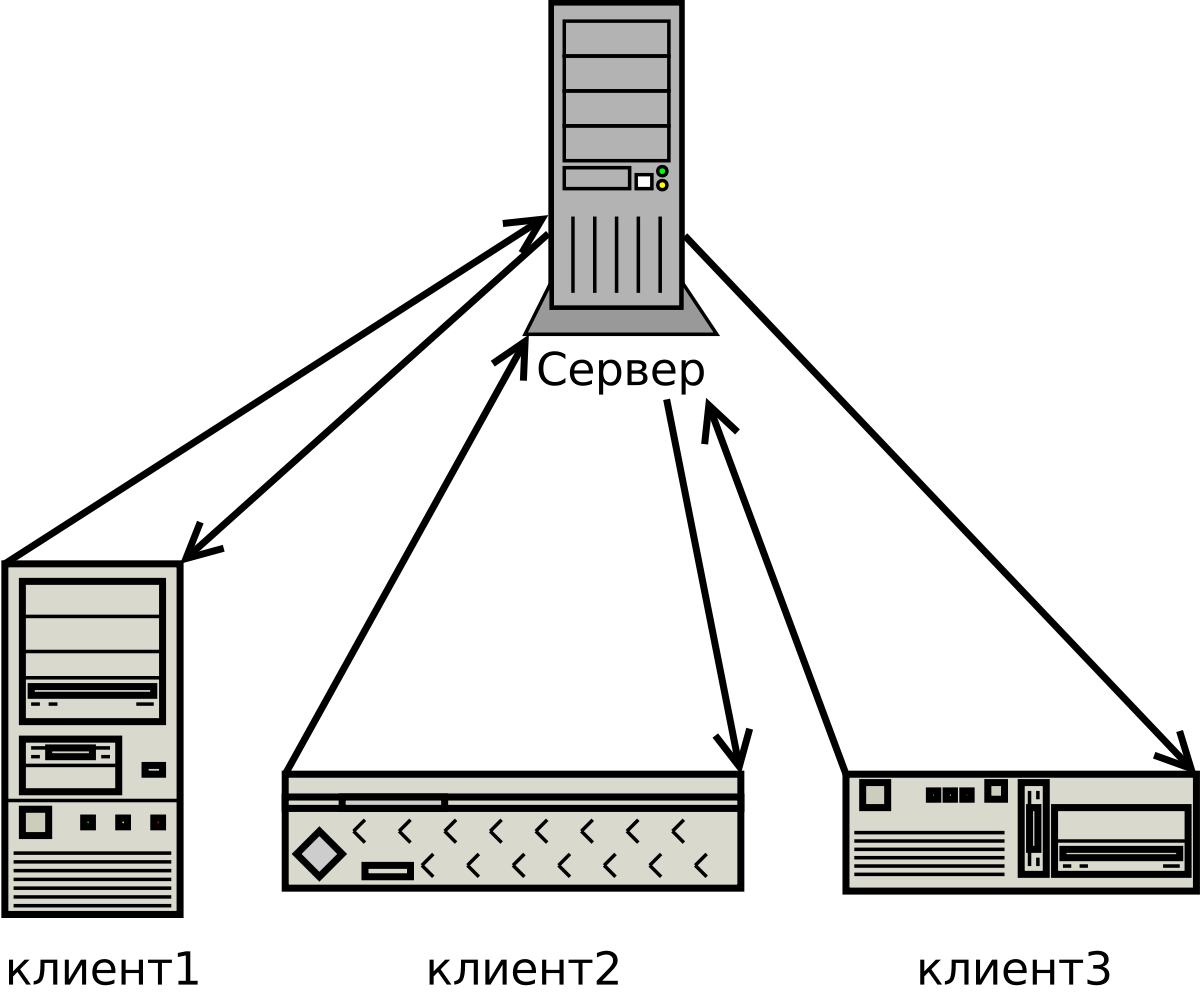


Рисунок 2.1 – Схема клиент-серверного приложения

Однако у такой системы существуют и недостатки. Например, недоступность сервера может сделать неработоспособной все вычислительную сеть.

Серверное приложение разработано с применением архитектура REST. REST (Representational state transfer) изначально был задуман как простой и однозначный интерфейс для управления данными, предполагавший всего несколько базовых операций с непосредственным сетевым хранилищем (сервером): извлечение данных (GET), сохранение (POST), изменение (PUT/PATCH) и удаление (DELETE). Разумеется, этот перечень всегда сопровождался такими опциями, как обработка ошибок в запросе, разграничение доступа к данным и валидация входящих данных, в общем, всеми возможными проверками, которые сервер выполняет перед тем, как выполнить желание клиента.

Помимо этого REST имеет ряд архитектурных принципов:

* независимость сервера от клиента — серверы и клиенты могут быть мгновенно заменены другими независимо друг от друга, так как интерфейс между ними не меняется. Сервер не хранит состояний клиента.  
  Уникальность адресов ресурсов — каждая единица данных имеет свой собственный уникальный URL, который, по сути, целиком является однозначным идентификатором ресурса. Пример: GET /api/v1/users/25/name
* независимость формата хранения данных от формата их передачи — сервер может поддерживать несколько различных форматов для передачи одних и тех же данных (JSON, XML и т.д.), но хранит данные в своем внутреннем формате, независимо от поддерживаемых.
* присутствие в ответе всех необходимых метаданных — помимо самих данных сервер должен возвращать детали обработки запроса, например, сообщения об ошибках, различные свойства ресурса, необходимые для дальнейшей работы с ним, например, общее число записей в коллекции для правильного отображения постраничной навигации.

Серверное приложение разработано с применением паттерна «Репозиторий» в уровне доступа к данным, который считается одним из наиболее часто используемых паттернов при работе с данными. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. Данный паттерн добавляет программе гибкость при работе с разными типами подключений.

База данных разрабатывалась с использованием ORM Sequelize, которая позволяет, при отсутствии базы данных, создать её на основе миграций и заполнить базу данных начальными данными при помощи сидов.

В качестве клиентского приложения было разработано Single Page Application. SPA -  это [веб-приложение](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), использующее единственный [HTML-документ](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) как оболочку для всех веб-страниц и организующий взаимодействие с пользователем через динамически подгружаемые [HTML](https://ru.wikipedia.org/wiki/HTML), [CSS](https://ru.wikipedia.org/wiki/CSS), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), обычно посредством [AJAX](https://ru.wikipedia.org/wiki/AJAX).

Плюсы SPA:

* SPA приложение отлично работают на устройствах как стационарных, так и мобильных. “Большие” компьютеры, планшеты, смартфоны, и, в конце-концов, простые телефоны могут беспрепятственно работать с сайтами построенных по принципу SPA. Работа на большом количестве устройств, а значит, создав одно приложение, вы получаете гораздо большую аудиторию пользователей нежели при использовании стандартного подхода;
* богатый пользовательский интерфейс, так называемый User Experience. Так как web-страница одна, построить богатый, насыщенный пользовательский интерфейс гораздо проще. Проще хранить информацию о сеансе, управлять состояниями представлений и управлять анимацией;
* SPA существенно сокращает так называемые “хождения по кругу”, то есть загрузку одного и того же контента снова и снова. Если ваш портал использует шаблон, то вместе с основным содержанием какой-либо страницы посетитель сайта обязательно загружает разметку шаблона.

Составляющие SPA:

* SPA поддерживает клиентскую навигации. Все “хождения” пользователя по модулям-страницам однозначно фиксируются в истории навигации, причем навигация при этом является “глубокой”, то есть если пользователь скопирует и откроет ссылку на внутреннюю модуль-страницу в другом браузере или окне, он попадет на соответствующую страницу.
* SPA размещается на одной web-странице, значит всё необходимое для работы сайта скрипты и стили должны быть определены в одном месте проекта – на единственной web-странице.
* SPA хранит постоянно состояние работы клиента в кэше браузера или в Web Storage.
* SPA загружает все скрипты требующиеся для старта приложения при инициализации web-страницы.
* SPA постепенно подгружает модули по требованию.

При реализации программы следует учесть разбиение функционала в зависимости от роли пользователя и дать каждому доступ к определенным функциям и ограничить доступ к иным. Данной ограничение реализовано, при помощи промежуточных обработчиков. Программа предусматривает разделение на 2 роли, а именно на администратора и обычного пользователя. Каждый из пользователей должен быть зарегистрирован, чтобы иметь возможность использовать приложение. Если это новый пользователь, то он может зарегистрироваться самостоятельно.

Авторизованный пользователь имеет следующий функционал:

* поиск заболеваний по определенному фильтру;
* проверка лекарственных средств на совместимость.

Администратор имеет следующий функционал:

* создание, редактирование, удаление заболевания;
* создание, редактирование, удаление группы лекарств;
* добавление лекарства в определенную группу;
* создание, редактирование, удаление рекомендаций;
* привязка лекарств на несовместимость.

Любые (в том числе и программные) системы проектируются с учетом того, что в процессе своей работы они будут использоваться людьми и/или взаимодействовать с другими системами. Поэтому основным и самым распространённым видом uml-диаграмм являются диаграммы вариантов использования (рисунок 2.2).

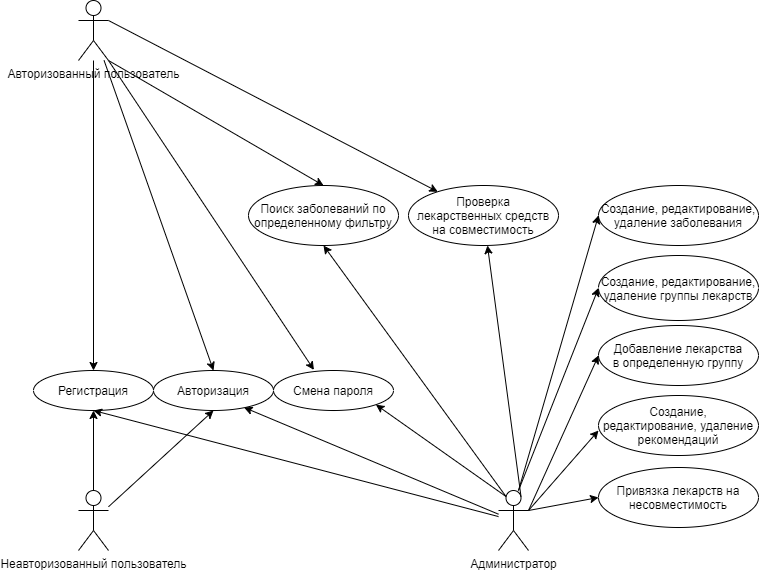


Рисунок 2.2 – Диаграмма вариантов использования

* 1. Архитектура системы

Под архитектурой программных систем будем понимать совокупность решений относительно:

* организации программной системы;
* выбора структурных элементов, составляющих систему и их интерфейсов;
* поведения этих элементов во взаимодействии с другими элементами;
* объединение этих элементов в подсистемы;
* архитектурного стиля, определяющего логическую и физическую организацию системы: статические и динамические элементы, их интерфейсы и способы их объединения.

Архитектура программной системы охватывает не только ее структурные и поведенческие аспекты, но и правила ее использования и интеграции с другими системами, функциональность, производительность, гибкость, надежность, возможность повторного применения, полноту, экономические и технологические ограничения, а также вопрос пользовательского интерфейса. Архитектура разрабатываемой системы представлена на рисунке 2.3

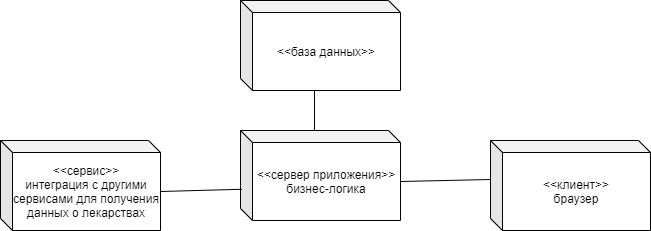


Рисунок 2.3 – Диаграмма архитектуры системы

Для разрабатываемого веб-приложения была выбрана классическая клиент-серверная архитектура, т.е. состоящая из сервера, узлов(сервисов) и клиентов.

Для разгрузки серверной части было принято решение отделить интеграцию со сторонними сервисами в отдельный сервис.

Так как в организациях уже развернута и налажена собственная сеть, внедрение системы с такой архитектурой не приведет к проблемам и материальным затратам.

2. 2. Информационная модель

Основу автоматизированной системы составляют базы данных. Широкое распространение получили реляционные (табличные) базы данных. Для безотказного пользования базой данных необходимо тщательное ее проектирование, состоящее в основном в определении состава таблиц и связей между ними.

Основные задачи проектирования БД:

* обеспечение надежного хранения в базе всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* исключение избыточности и дублирования данных;
* обеспечение целостности базы данных.

В процессе проектирования базы данных для разрабатываемого программного комплекса были выполнены следующие этапы:

* уточнение задач: выявление задач системы, требующих коммуникаций с БД, выделение их в группы;
* анализ данных: составление подробного перечня всех данных, необходимых для решения каждой задачи;
* определение структуры данных: упорядочивание данных по объектам и определение связей между ними, или нормализация;
* тестирование в результате выполнения запросов к базе данных и ее совершенствование.

Последний этап может проводился в процессе разработки приложения. Схема используемой базы данных представлена на рисунке 2.4.

Были выделены следующие сущности:

* сущность «Заболевание» - хранит всю информацию о заболеваниях;
* сущность «Лекарство» - хранит информацию о лекарствах, которые были получены со сторонних сервисов;
* сущность «Группа» - хранит информацию о лекарственных группах;
* сущность «Порядок отпуска» - хранит информацию о видах отпуска лекарственных средств;
* сущность «Рекомендация» - хранит информацию о рекомендациях по лечению определенных заболеваний;
* сущность «Путь введения» - хранит информацию о методах введения лекарственного средства;
* сущность «Симптом» - хранит информацию о различных симптомах при заболевании;
* сущность «Тип лекарства» - хранит информацию о типах лекарственных средств;
* сущность «Пользователь» - хранит информацию о пользователях.

Физическая модель базы данных определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Построение физической модели базы данных производится на основе логической модели, которая и была представлена выше.

Таблица 2.1 – Схема таблицы заболевания (Illnesses)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Название заболевания |
| description | VARCHAR | Подробное описание заболевания |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.2 – Схема таблицы лекарства (Medicines)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Название лекарства |
| description | TEXT | Подробное описание лекарства |
| wayUse | TEXT | Инструкция по применению лекарства |
| typeMedicineId | INT | Идентификатор типа лекарства |
| routeId | INT | Идентификатор пути ввода лекарства |
| productionMedicineMethodId | INT | Идентификатор порядка отпуска лекарства |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.3 – Схема таблицы группы (Groups)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Наименование группы |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.4 – Схема таблицы порядок отпусков (ProductionMedicineMethods)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Наименование порядка отпуска |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.5 – Схема таблицы рекомендации (Recommendations)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Наименование рекомендации |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.6 – Схема таблицы пути ввода (Routes)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Наименование пути ввода |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.7 – Схема таблицы симптомы (Symptoms)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Наименование симптома |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.8 – Схема таблицы типы лекарств (Typemedicines)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| name | VARCHAR | Наименование типа лекарства |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

Таблица 2.9 – Схема таблицы пользователи (Users)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | INT | Целочисленный идентификатор, значение устанавливается при помощи автоматической инкрементации |
| firstname | VARCHAR | Имя пользователя |
| lastname | VARCHAR | Фамилия пользователя |
| email | VARCHAR | Электронная почта пользователя |
| password | VARCHAR | Пароль пользователя |
| createdAt | DATETIME | Дата создания записи |
| updatedAt | DATETIME | Дата последнего обновления записи |

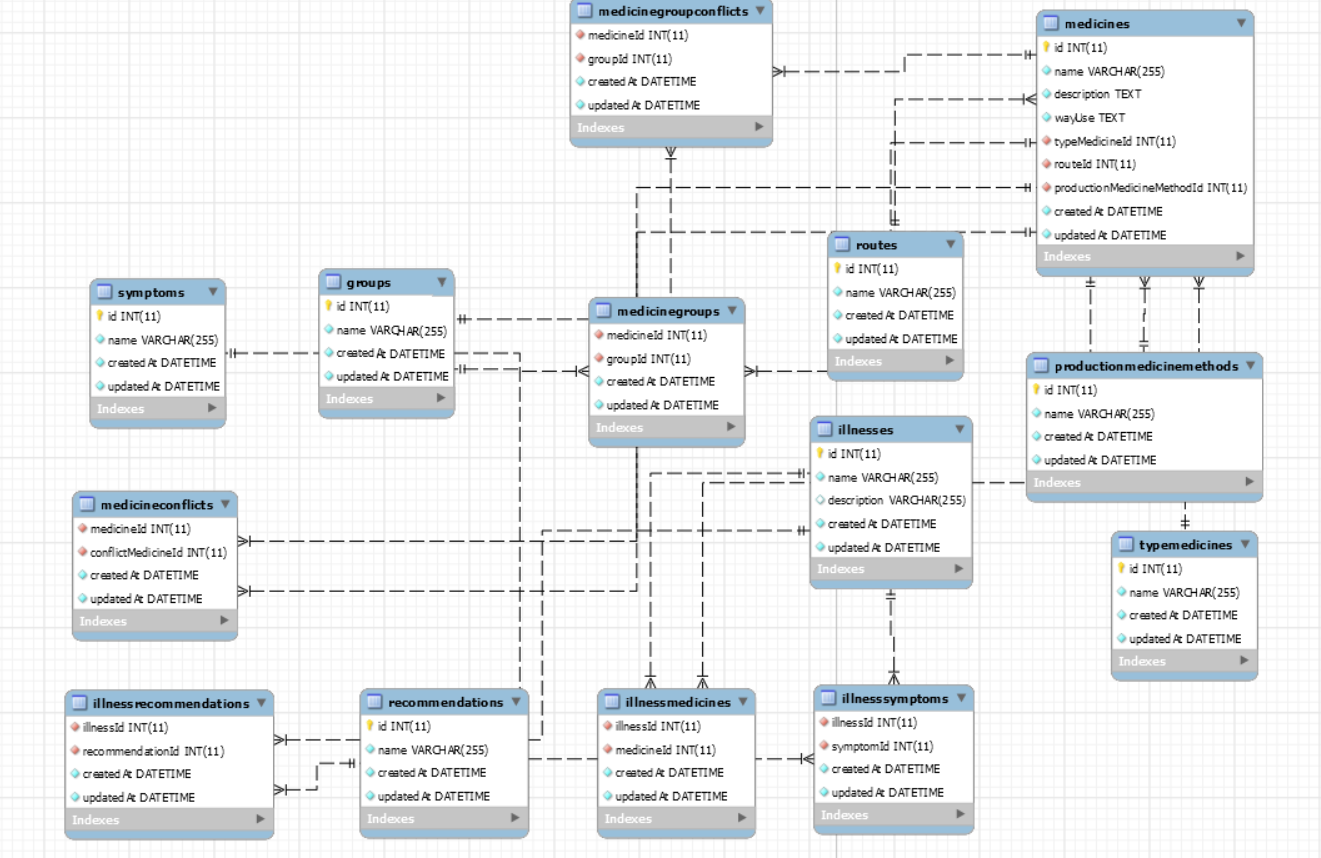


Рисунок 2.4 – ER-диаграмма базы данных

# **ЗАКЛЮЧЕНИЕ**

В ходе преддипломной практики было положено начало работы над дипломной работой, то есть собрана и анализирована информация по предметной области, выявлены ее проблемы и пути их решения. Собранная информация является основой для первых двух глав пояснительной записки.

Как результат, есть в наличии функциональная структура разрабатываемой системы в виде uml-диаграмм, а также диаграмма архитектуры системы, которые положат начало написания программного кода и сделает дальнейшую разработку более прозрачной и последовательной.

За время прохождения практики все поставленные задачи были выполнены успешно.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Node.js – [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/nodejs/1.1.php> - Дата доступа: 19.04.2020.
2. Sequelize – [Электронный ресурс]. – Режим доступа: http://docs.sequelizejs.com/ – Дата доступа: 15.04.2020.
3. [СовременныйучебникJavaScript](https://learn.javascript.ru/) – [Электронный ресурс]. – Режим доступа: <https://learn.javascript.ru/>, – Дата доступа: 19.04.2020.
4. React – [Электронный ресурс] – Режим доступа: http://docs.reactjs.com/ – Дата доступа: 15.04.2020.