Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра: Экономической информатики

Факультет: ИТУ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему

«Автоматизированная система покупки одежды»

Выполнил: студент группы 524402

Лащётко Антон Владимирович

Руководитель: Петрович Никита Олегович

Минск 2017

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ](#_Toc481447850) 5

[1. ОПИСАНИЕ ПРОЦЕССА ПОКУПКИ ТОВАРА](#_Toc481447851) 7

[1.1 Интернет-магазин как современное средство торговли](#_Toc481447851) 7

[1.2 Android](#_Toc481447851) 8

[1.3](#_Toc481447851)

[2. ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ.](#_Toc481447852)

[2.1 JavaScript](#_Toc481447851)

[2.2 React Native](#_Toc481447851)

[2.3 Express](#_Toc481447851)

[2.4 MongoDB](#_Toc481447851)

[2.5 GraphQL](#_Toc481447851)

[2.6 Visual Studio Code](#_Toc481447851)

[3. ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ](#_Toc481447853)

[4. ОПИСАНИЕ ИНФОРМАЦИОННОЙ МОДЕЛИ СИСТЕМЫ](#_Toc481447854)

[5. МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ](#_Toc481447855)

[6. ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ И ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРЕКТИРОВАНИЯ](#_Toc481447856)

[7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ](#_Toc481447857)

[8. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ](#_Toc481447858)

[ЗАКЛЮЧЕНИЕ](#_Toc481447859)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc481447860)

[ПРИЛОЖЕНИЕ А UML модели](#_Toc481447861)

[ПРИЛОЖЕНИЕ Б Блок схемы алгоритмов](#_Toc481447861)

[ПРИЛОЖЕНИЕ В Листинг кода](#_Toc481447861)

[ПРИЛОЖЕНИЕ Г Листинг запросов](#_Toc481447861)

**ВВЕДЕНИЕ**

Выбранная тема считается актуальной на сегодняшний день, так как сегодня миллионы людей ежедневно, не выходя из дома, покупают различные товары в электронных магазинах. В мире огромными темпами растет количество пользователей интернет и, как следствие, количество «электронных» покупателей и потенциальных «электронных» покупателей.

Использование электронных технологий в интернете во всех сферах жизни является одной из существенных черт и предпосылок формирования информационного общества и процесса глобализации мировой экономики. Интернет открыл новые возможности и формы экономических связей между участниками движения товаров, ресурсов и денег. В экономике развитых стран получили бурное развитие различные формы электронного бизнеса и, в частности, его важнейшего сегмента - электронной торговли.

Электронные магазины существенно уменьшают издержки производителя, сэкономив на содержании обычного магазина, расширяют рынки сбыта, так же, как и расширяет возможность покупателя – покупать любой товар в любое время в любой стране, в любом городе, в любое время суток, в любое время года. Это дает электронным магазинам неоспоримое преимущество перед обычными магазинами. Этот момент является существенным при переходе производителей с «обычной» торговли на «электронную».

Целью данной курсовой работы является предоставление пользователям удобного мобильного сервиса, с помощью которого пользователь, без особого труда, сможет выбрать и купить желаемый товар, не выходя из дома.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить предметную область;
* проанализировать логическую и физическую модель представления данных;
* создать базу данных;
* реализовать серверную часть, которая позволит работать с базой данных и выполнить ряд функций, поставленных целью разработки курсового проекта;
* разработать программное обеспечение;
* протестировать программный продукт.

Целевой платформой данного курсового проекта будут мобильные устройства с операционными системами Android, так как в современном обществе практически у каждого есть смартфон. Это открывает доступ к огромной базе покупателей, что важно, особенно для магазинов.

**1. ОПИСАНИЕ ПРОЦЕССА ПОКУПКИ ТОВАРА**

* 1. **Интернет-магазин как современное средство торговли**

Человеческие потребности быть всегда в курсе дел выводят информационные технологии на прямую линию по созданию все новых девайсов и гаджетов. Неудобство эксплуатации компьютеров и ноутбуков обуславливает появление различных мини-компьютеров, смартфонов и коммуникаторов, в основе которых лежит все та же операционная система. Благодаря техническому прогрессу, чуть ли не ежедневно появляется масса новейших разработок в области мобильных технологий. Иногда это происходит так быстро, что порой бывает сложно уследить за всеми новыми технологиями.

Использование электронных технологий и интернете во всех сферах жизни является одной из существенных черт и предпосылок формирования информационного общества и процесса глобализации мировой экономики.

С течением времени изменилось многое: Современный Интернет - это огромное хранилище, в котором каждый по своему желанию может найти нужный текст, музыку, учебные материалы, а с некоторого времени и просто купить все, что необходимо - от продуктов, до программного обеспечения. В данное время многие дальновидные руководители компаний совершенно справедливо считают необходимым создание небольшого Интернет-представительства компании, другими словами все больше появляется коммерческих сайтов, основной целью которых является продвижение товаров и услуг с помощью Интернет.

Интернет открыл новые возможности и формы экономических связей между участниками движения товаров, ресурсов и денег. В экономике развитых стран получили бурное развитие различные формы электронного бизнеса и, в частности, его важнейшего сегмента - электронной торговли.

Интернет магазин, как важнейший элемент электронной торговли, - это магазин, «витрина» которого расположена в интернете и который дает возможность заказать товар через интернет.

Создание качественной системы Internet-торговли невозможно без изучения посетителей. Для этого используются различные технологии отслеживания, с помощью которых продавец узнаёт, какие товары выбирает каждый из посетителей. Эти сведения используются также в сфере индивидуального маркетинга и бизнеса, основу которого составляют хранимые предпочтения отдельных потребителей.

Главным условием успешной работы Интернет-магазина является учет разнообразных потребностей покупателей. Именно этот фактор определяет организационную структуру предприятия, а также состав и содержание осуществляемых бизнес-процессов.

Виртуальный магазин будет привлекателен для потенциального покупателя только в том случае, если он сможет заинтересовать его предлагаемыми товарами и услугами по приемлемым ценам и надлежащего качества. Поэтому ключевыми задачами Интернет-магазина являются формирование оптимального ассортимента, выбор правильной ценовой политики и организация эффективной работы с поставщиками.

Для магазина одежды — это также важно. Одежда в современном мире является показателем статности, и нередко люди готовы потратить большие деньги, чтобы купить ту или иную вещь.

Для того, чтобы пользователь получил право на покупку товара, ему необходимо пройти идентификацию. При использовании метода базовой идентификации пользователи регистрируются в приложении с помощью идентификатора и пароля, по которым сервер и опознает посетителей.

Также, необходимо продумать систему обработки заказов. Здесь ключевую роль играет набор функций виртуальной корзины, в которую потребитель «складывает» все выбранные им товары. Среди этих функций обязательно должны присутствовать такие, которые позволяли бы потребителю добавлять позиции (товары) в корзину, удалять позиции, а также изменять количество единиц товара в корзине. После того как клиент завершил отбор товаров, ему предлагается список, в который он может внести последние изменения и который он передает на контроль (в «кассу»), где происходит оплата покупок. На программном уровне электронный магазин должен поддерживать несколько технологий оплаты, среди которых - оплата по кредитной карте с помощью протоколов SSL.

**2.2 Android**

Но в современном мире не все хотят открывать браузер на мобильных устройствах, а тем более искать удобные сайты, которые, к тому же, в основном, спроектированы под большие мониторы компьютеров и на различных экранах мобильных девайсов будут смотреться по-разному.

Если раньше можно было звонить только по стационарному проводному телефону, получить доступ в сеть только с персонального компьютера, то сейчас никого не удивляют мобильные аппараты с беспроводным Интернетом, GPS-навигацией и модулем Wi-Fi. Новые мобильные технологии распространяются невероятно быстрыми темпами. Если новинки мобильных технологий будут и дальше появляться с такой частотой, некоторые из них можно просто не заметить. Лидирующие позиции среди мобильных смартфонов на сегодняшний день занимают платформы Android и iPhone. У мобильных технологий многообещающая перспектива, и Android представляет собой жизнеспособную и интересную для разработчиков платформу. Android-устройства набирают популярность быстрее, чем любая другая мобильная платформа. Оборотная сторона массового принятия Android - это фрагментация, которая создает значительные проблемы для тестирования приложений. Для большинства разработчиков реальность такова, что почти невозможно протестировать приложение для каждой комбинации Android-устройства и версии ОС. Приложения под операционную систему Android разрабатываются в основном с использованием Java, однако существуют библиотеки, позволяющие вести разработку на языке С++. Скомпилированный программный код упаковывается в специальный файл-архив, AndroidPackage. Этот файл имеет расширение \*.apk и упаковывается специальной утилитой aapttool. Именно он в дальнейшем распространяется как программа и инсталлируется на мобильные устройства. Один такой файл связан с кодом одного приложения. И каждое приложение в Android живет в своем собственном мире - в такой машине. По умолчанию, каждая программа выполняется в своем собственном процессе, управлением которого занимается ядро Linux, которое также осуществляет менеджмент памяти. Таким образом, чаще всего код приложения выполняется в изоляции от всех других приложений. Android стартует процесс, когда возникает необходимость выполнить какой-нибудь программный код и завершает его, когда в нем больше нет необходимости и системные ресурсы требуются другим приложениям.

По умолчанию, каждому приложению присваивается свой уникальный ID Linux-пользователя. Права доступа устанавливаются таким образом, чтобы файлы приложения были видны только этому пользователю и данному приложению. Хотя, есть способы, позволяющие экспортировать их в другие приложения. Например, существует возможность «разделения» одного и того же пользовательского ID между двумя приложениями. В таком случае, они смогут видеть файлы друг друга. Для того, чтобы экономить системные ресурсы, приложения с одинаковым ID можно также договориться запускать в одном и том же Linux-процессе, разделяя одну и ту же виртуальную машину.

Интересной чертой операционной системы Android является тот факт, что приложение в ней может использовать для своего функционала элементы других приложений, если, конечно, они предоставляют такой доступ. Например, если нашему приложению требуется отобразить прокручиваемый список изображений, а другое приложение уже имеет реализованный подходящий скроллер, открытый для других приложений, то можно просто вызвать его для осуществления этого действия и не разрабатывать свой собственный. При этом код нашего приложения не смешивается с кодом другого и не компонуется с ним. Скорее, он просто запускает некоторый кусок другого файла, когда возникает такая необходимость. Для обеспечения такого принципа работы, система должна быть способна запускать процесс приложения тогда, когда потребуется любая его часть и создавать экземпляры java-объектов именно для этой части. Поэтому приложения Android не имеют единой точки входа, как это принято в большинстве систем. Вместо этого их код представляет собой набор некоторых отдельных целостных сущностей, компонент, из которых система по мере необходимости может создавать экземпляры и использовать их.

В процессе пользования мобильным интернет-магазином пользователь получит удобный интерфейс, с помощью которого сможет просматривать товары без каких-либо неудобств в любом месте в любое время. Пользователь также получает удобный сервис покупки товаров, а администратор – точно такой же мобильный, гибкий функционал, с помощью которого он сможет просматривать приложение с позиции пользователя и, одновременно с этим, редактировать товары с помощью своего мобильного устройства.

**2. ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ.**

Задачей курсового проекта является создание клиент-серверного мобильного программного приложения, позволяющего осуществить облегчённый процесс покупки-продажи одежды.

**2.1 JavaScript**

Программный код должен быть написан на языке JavaScript.

Язык JavaScript очень популярен благодаря присутствию в любом веб-браузере. Он ни в чем не уступает другим языкам, но при этом поддерживает многие современные представления о том, каким должен быть язык программирования. Это динамический язык со слабо типизированными, динамически расширяемыми объектами, которые неформально объявляются по мере необходимости. Функции в нем являются полноценными объектами и обычно используются в виде анонимных замыканий. Это делает JavaScript более мощным языком, по сравнению с некоторыми другими, часто применяемыми для разработки веб-приложений.

Один из основных недостатков JavaScript - Глобальный Объект. Все переменные верхнего уровня «сваливаются» в Глобальный Объект, и, при использовании одновременно нескольких модулей, это может привести к неуправляемому хаосу.

**2.2 React Native**

Для моего приложения, клиентская часть написана на фреймворке React Native.

React Native — это JS-фреймворк для создания нативно отображаемых iOS- и Android-приложений. В его основе лежит разработанная в Facebook JS-библиотека React, предназначенная для создания пользовательских интерфейсов, а также кроссплатформенных приложений для iOS и Android. Он появился в начале 2015 года, не использует WebView и HTML-технологии. React Native современный и идеально подходит для написания мобильных приложений.

**2.3. Express**

Серверная часть написана на Express.js.

Express.js, или просто Express, каркас web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описывает его как созданный на основе написанного на языке Ruby каркаса Sinatra, подразумевая, что он минималистичен и включает большое число подключаемых плагинов.

Преимущества клиент-серверных систем:

* клиент-серверный подход модульный, причем серверные программные компоненты компактны и автономны;
* сбой сервера не повлияет на остальные компоненты операционной системы;
* автономность компонентов делает возможным их выполнение на нескольких процессорах на одном компьютере (симметричная многопроцессорная обработка) или на нескольких компьютерах сети (распределенные вычисления);
* обязанность клиента, как правило, – предоставлять пользовательские сервисы и, прежде всего, пользовательский интерфейс, то есть средства для приема, отображения и редактирования данных, введенных пользователем, которые служат основой для запроса серверу.

Связь между клиентом и сервером осуществляется с помощью HTTP. HTTP (HyperText Transfer Protocol, «протокол передачи гипертекста») – широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

**2.4. MongoDB**

База данных – MongoDB.

MongoDB – документно-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Я решил использовать именно MongoDB потому, что эта СУБД идеально подходит под Express и использует один из наиболее удобных форматов данных при взаимодействии с JavaScript – JSON. Если нужно с сервера взять объект с данными и передать его клиенту, то в качестве промежуточного формата – для передачи по сети, почти всегда используют именно его.

**2.5. GraphQL**

Для запроса данных с сервера я решил использовать GraphQL.

GraphQL это язык запросов, который описывает как запрашивать данные, и, в основном, используется клиентом для загрузки данных с сервера. Он предоставляет полное и понятное описание данных, а также дает клиентам возможность запрашивать именно то, что им нужно, и не более того. Также, его использование облегчает формирование единого состояния (state) для react-приложений.

GraphQL имеет три основные характеристики:

* Позволяет клиенту точно указать, какие данные ему нужны;
* Облегчает агрегацию данных из нескольких источников;
* Использует систему типов для описания данных.

**2.6. Visual Studio Code**

В качестве основной среды разработки используется Visual Studio Code.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Основными преимуществами, из-за которых и была выбрана эта среда, являются:

* Поддержка git;
* Многооконность;
* Поддержка различных расширений, удобных для разработки мобильных приложений;
* Встроенный многооконный терминал.

**3. ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ**

После проведенного анализа предметной области представим функциональную модель процесса пользования интернет магазином одежды.

Входными данными, необходимыми для выполнения процесса являются:

* База данных одежды;
* База данных пользователей;
* Пожелания клиента.

Ресурсами, необходимыми для выполнения процесса являются:

* Пользователь;
* Администратор;
* Оборудование.

Основным процессом данного курсового проекта является покупка товара пользователем. В соответствии с этим на рисунке 3.1 представлена контекстная диаграмма функциональной модели.

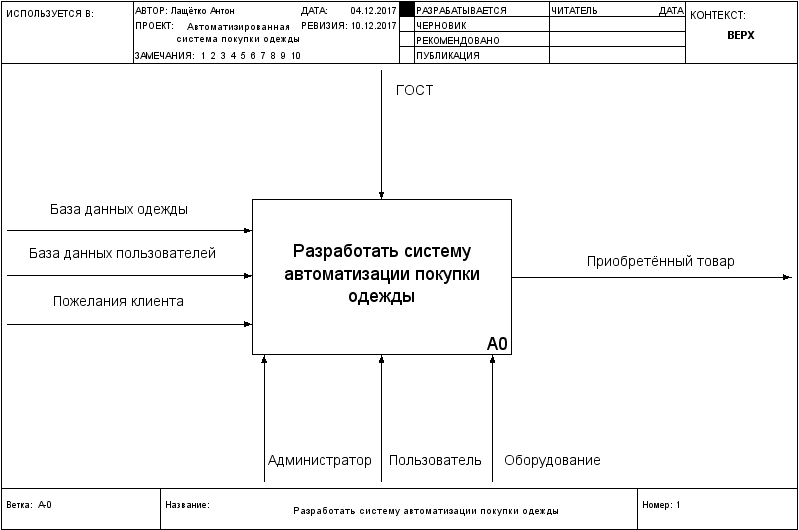


Рисунок 3.1 Контекстная диаграмма верхнего уровня.

Декомпозиция контекстной диаграммы, показанная на рисунке 3.2, описывает разделение процесса покупки товара на следующие этапы:

* Зарегистрировать пользователя;
* Добавить товар в корзину;
* Оплатить товар.

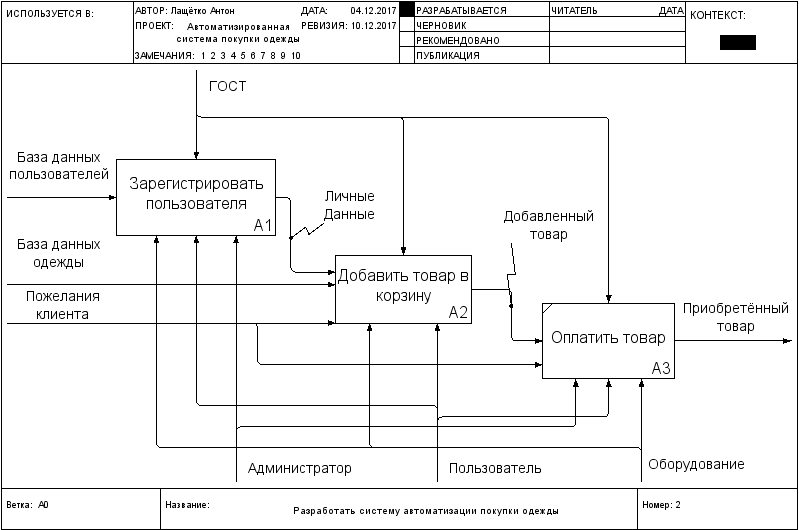


Рисунок 3.2 – Декомпозиция диаграммы верхнего уровня.

Декомпозиция блока «Зарегистрировать пользователя» (см. рис. 3.3). Данный блок можно разбить следующим образом:

* Ввести имя;
* Ввести почту;
* Ввести номер телефона;
* Ввести пароль.

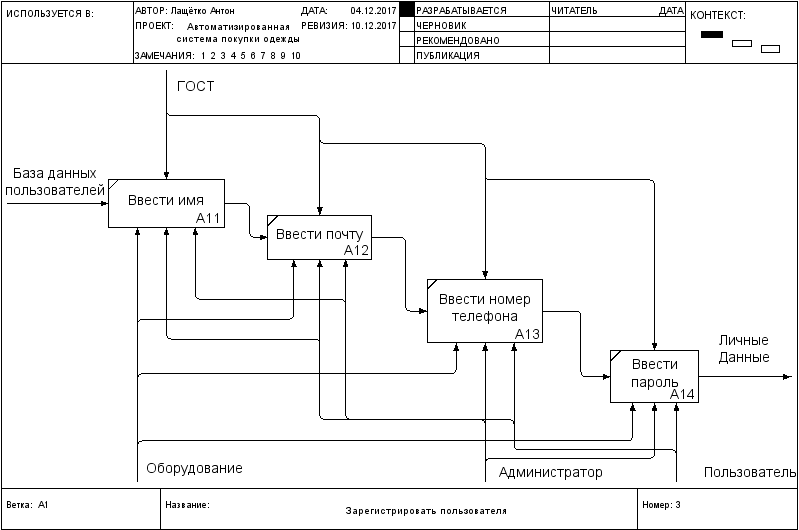


Рисунок 3.3 – Декомпозиция блока «Зарегистрировать пользователя».

Декомпозиция блока «Добавить товар в корзину» (см. рис. 3.4). Данный блок можно разбить следующим образом:

* Выбрать фильтры;
* Добавить товар.

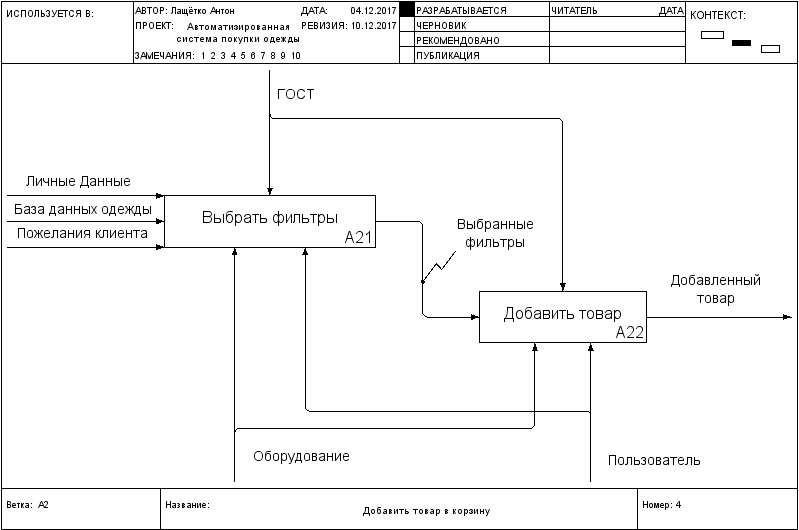


Рисунок 3.4 – Декомпозиция блока «Добавить товар в корзину».

Следующей идёт декомпозиция блока «Выбрать фильтры» (см. рис. 3.5). Данный блок можно разбить следующим образом:

* Выбрать пол;
* Выбрать тип одежды.

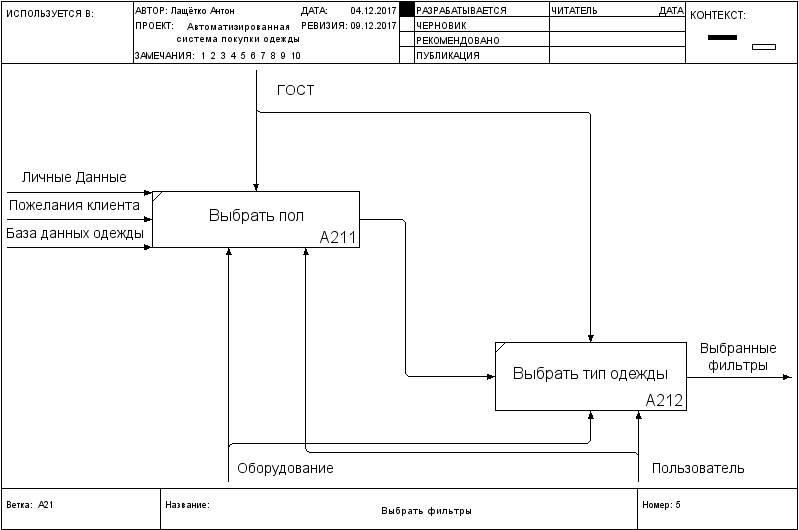


Рисунок 3.5 – Декомпозиция блока «Выбрать фильтры».

На одном уровне с предыдущей идёт декомпозиция блока «Выбрать товар» (см. рис. 3.6). Данный блок можно разбить следующим образом:

* Выбрать товар;
* Выбрать размер;
* Добавить в корзину.

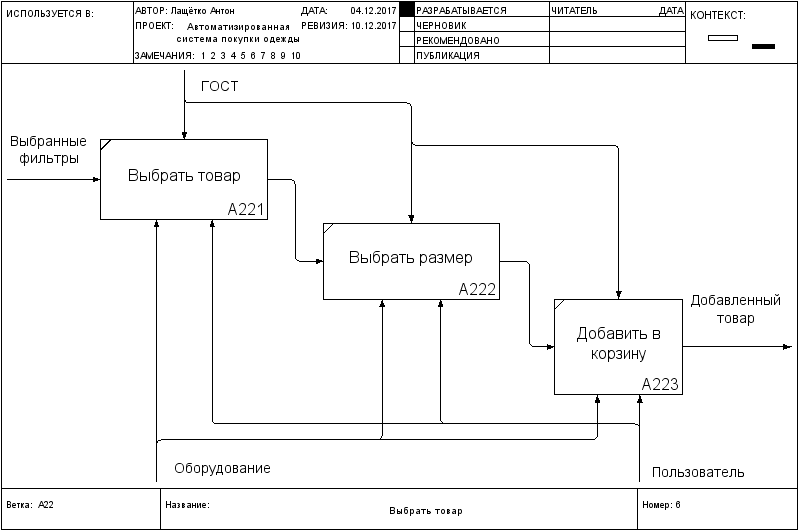


Рисунок 3.6 – Декомпозиция блока «Выбрать Товар»

С помощью представленной IDEF-0 диаграммы можно более подробно и тщательно рассмотреть процесс покупки одежды.

# **4. ОПИСАНИЕ ИНФОРМАЦИОННОЙ МОДЕЛИ СИСТЕМЫ**

В результате информационного моделирования (см. рис. 4.1) были выделены следующие сущности:

* Пользователи;
* Одежда.

Для хранения личной информации об пользователях необходимо

использовать сущность «Пользователи».

Данная таблица состоит из нескольких полей, таких как:

* Username – имя пользователя. Необходимо для удобного обращения к пользователю;
* Password – пароль пользователя. Необходим для идентификации пользователя в совокупности с полем email;
* isAdmin – поле, показывающее является ли пользователь администратором;
* Email – почта пользователя. Необходима для идентификации пользователя в совокупности с password;
* Favourites – поле, содержащее в себе идентификаторы одежды, которые пользователь добавил в любимые;
* Basket – поле, содержащее в себе идентификаторы одежды, которые пользователь добавил в корзину;
* Mobile – мобильный телефон пользователя. Необходим для связи с пользователем.

Таблица «Одежда» содержит информацию об одежде. Она состоит из полей, таких как:

* Id – уникальный идентификатор. Необходим для поиска и идентификации одежды;
* Name – название одежды;
* Price – цена одежды;
* Sizes – размеры одежды;
* Images – изображения одежды;
* Filter – фильтр пола, по которому, при соответствии, будет сортироваться одежда. Принимает значения: “Men”, “Women”, “Junior”;
* Topic – фильтр типа одежды, по которому, при соответствии, будет сортироваться товар. Принимает значения: “Jacket”, “Jumper”, “Shoes”, “Pants”.

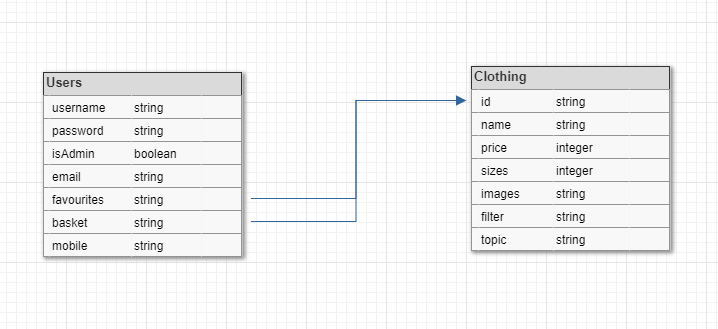


Рисунок 4.1 – Информационная модель базы данных.

С учетом обозначенного взаимодействия сущностей смоделируем их взаимодействие в формате IDEF1.X и приведем эту модель к третьей нормальной форме. В результате последовательного приведения модели к третьей нормальной форме, получаем информационную модель, соответствующую условиям третьей нормальной формы – не ключевой атрибут сущности функционально зависит только от всего первичного ключа и ни от чего другого.

# **5. МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

Для более подробного изучения и рассмотрения системы программы необходимо рассмотреть такие модели как:

* диаграмма прецедентов;
* диаграмма последовательностей;
* диаграмма классов;
* диаграмма развертывания;
* диаграмма компонентов;
* диаграмма состояний.

Система управления работой будет реализована на операционной системе Android. На диаграмме развертывания (рисунок 5.1), показана конфигурация обрабатывающих узлов, на которых выполняется система, и компонентов, размещенных в этих узлах.

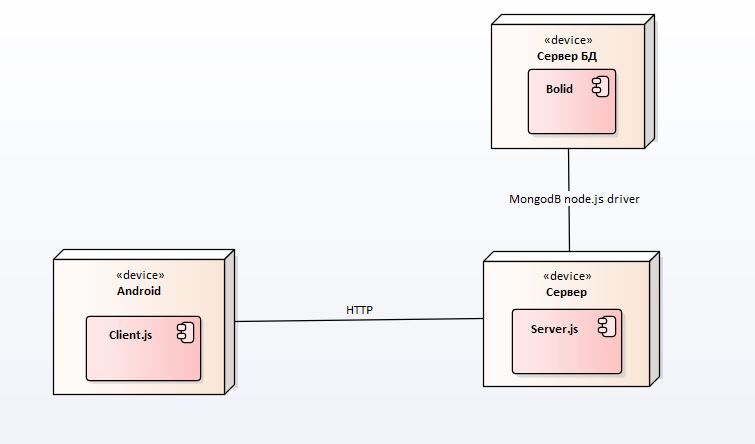


Рисунок 5.1 – Диаграмма развёртывания.

Для входа в систему, пользователю требуется ввести логин и пароль. После выполнения необходимих действий, клиент отправляет запрос серверу, а сервер уже сверяет введенные данные с БД. В БД формируется ответ серверу, который, в свою очередь, формирует из него ответ для клиента. Клиентская часть предоставляет результат авторизации пользователю (приложение А.1).

Диаграма прецедентов (рис. 5.3) – это своеобразный инструмент для поиска и проверки всех возможных сценариев. На каждом шаге происходит проверка удачного, или неудачного развития событий. Таким образом, с помощью этой диаграммы, можно с наименьшими тратами времени продумать все пути развития всех возможных сценариев.

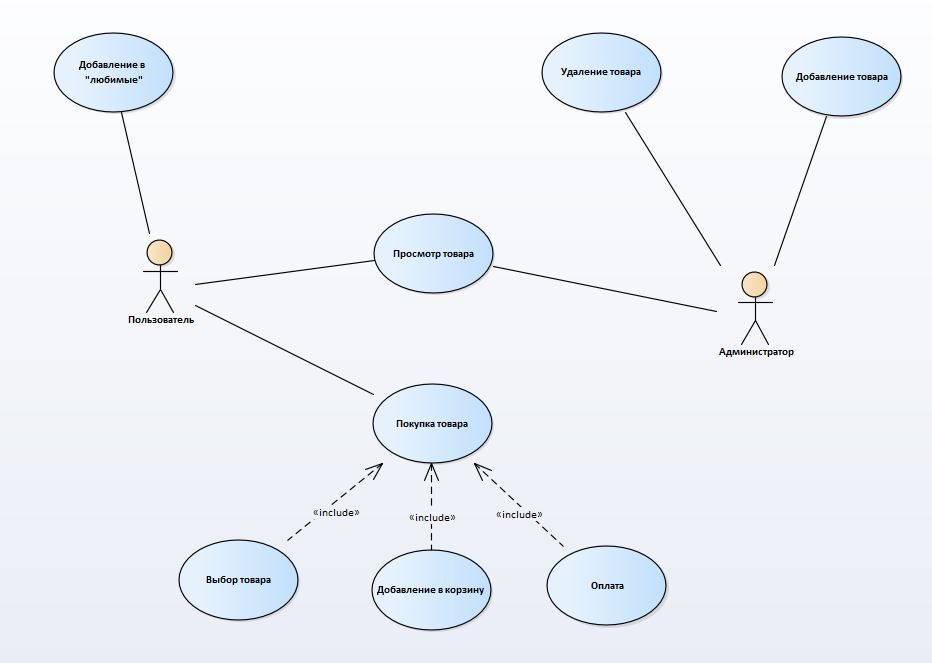


Рисунок 5.3 – Диаграмма прецедентов.

Полный проект программной системы представляет собой совокупность моделей разных уровней, которые должны быть согласованы между собой. Диаграмма компонентов отражает общие зависимости между компонентами. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код. Диаграмма компонентов для данного проекта представлена ниже (рис. 5.4).

В разделе Client находятся компоненты клиентской части системы. То, как будет выглядеть то или иное окно, определяется именно здесь. В разделе Server находятся компоненты серверной части системы. Здесь осуществляется запуск сервера, соединение с базой данных, обмен данных с клиентом.

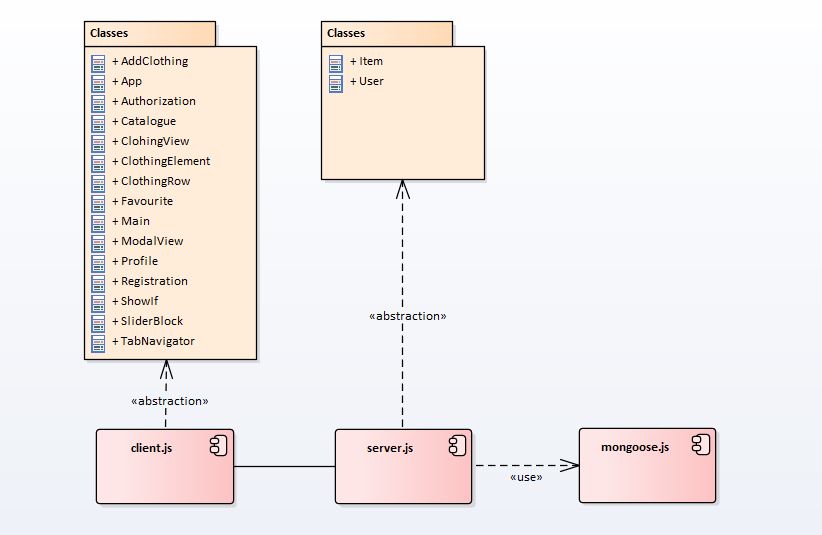


Рисунок 5.4 – Диаграмма компонентов.

Диаграмма классов (рис. 5.5) – это набор статических, декларативных элементов модели. Диаграммы классов могут применяться и при прямом проектировании, то есть в процессе разработки новой системы, и при обратном проектировании – описании существующих и используемых систем. Диаграмма классов – это конечный результат проектирования и отправная точка процесса разработки.

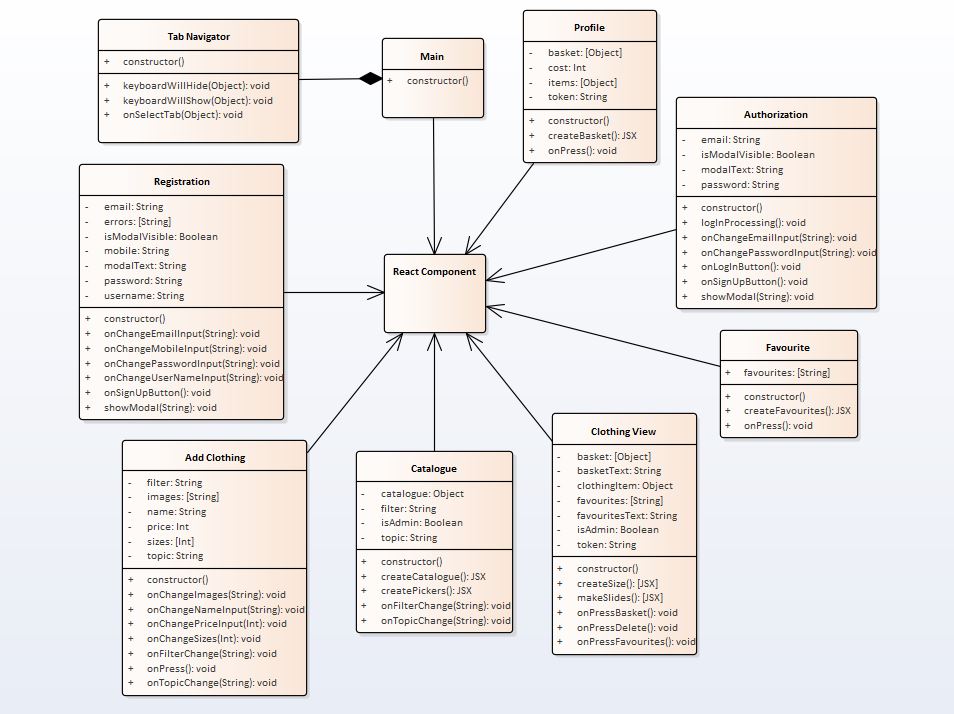


Рисунок 5.5 – Диаграмма классов.

Диаграмма состояний при добавлении товара представлена на рисунке А.2 приложения А.

**6. ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ И ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРЕКТИРОВАНИЯ**

Проверка логина и пароля реализована следующим образом:

checkUser: async function checkUser(\_, { email, password }, ctx) {

let userData;

const user = new ctx.constructor.User();

await user.findUser(email, password)

.then((data) => {

userData = data;

})

.catch(e => console.log(e));

if (userData) {

return {

token: `${userData.username}\_${userData.username.length}`,

message: 'Log in success',

username: `${userData.username}`,

isAdmin: userData.isAdmin,

favourites: userData.favourites,

basket: userData.basket,

};

}

return { message: 'Log in failed' };

}

Схема алгоритма процесса авторизации представлена в приложении Б на рисунке Б.1.

# 

# **7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Данное программное приложение представляет собой автоматизированную систему покупки и продажи одежды. Приложение включает в себя два интерфейса: интерфейс пользователя и администратора. У администратора появляются дополнительные возможности.

При входе в приложение пользователю представляется главная страница приложения, на которой находятся элементы, с помощью которых можно быстро перейти в каталог с выбранным фильтром, а также случайные 10 товаров. (рис. 7.1)

Рисунок 7.1 – Главное меню.

Для того, чтобы были доступны все функции пользователю нужно авторизоваться. Для этого нужно нажать на панели навигации на иконку профиля, после чего откроется экран авторизации (рис. 7.2)

Рисунок 7.2 – Экран авторизации.

Если у пользователя нет аккаунта в приложении – ему нужно перейти на экран регистрации (рис. 7.3). Для этого нужно нажать на кнопку Sign Up.

Рисунок 7.3 – Экран регистрации.

После авторизации пользователь попадает в корзину, в которой либо находятся товары, добавленные в неё, либо уведомление о том, что корзина пуста и кнопка для перехода в каталог (рис. 7.4 и 7.5)

Рисунок 7.4 – Корзина с элементами.

Рисунок 7.5 – Корзина без элементов.

После клика на любой из товаров открывается экран просмотра товара, который отличается у пользователя и администратора наличием у второго кнопки «Удалить товар» (рис. 7.6).

Рисунок 7.6 – Экран просмотра товара.

Каталог также отличается у пользователя и администратора наличием у второго кнопки «Добавить товар» (рис. 7.7).

Рисунок 7.7 – Экран каталога.

Вывод по разделу: в данном разделе была полностью рассмотрена функциональная сторона приложения.

# **8. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ**

В ходе выполнения курсового проекта было разработано клиент-серверное приложение покупки одежды.

Необходимо проверить правильно проверки на авторизацию и регистрацию пользователя. Если пользователь неправильно ввёл какие-то из данных – система выдаст предупреждение (рис. 8.1).

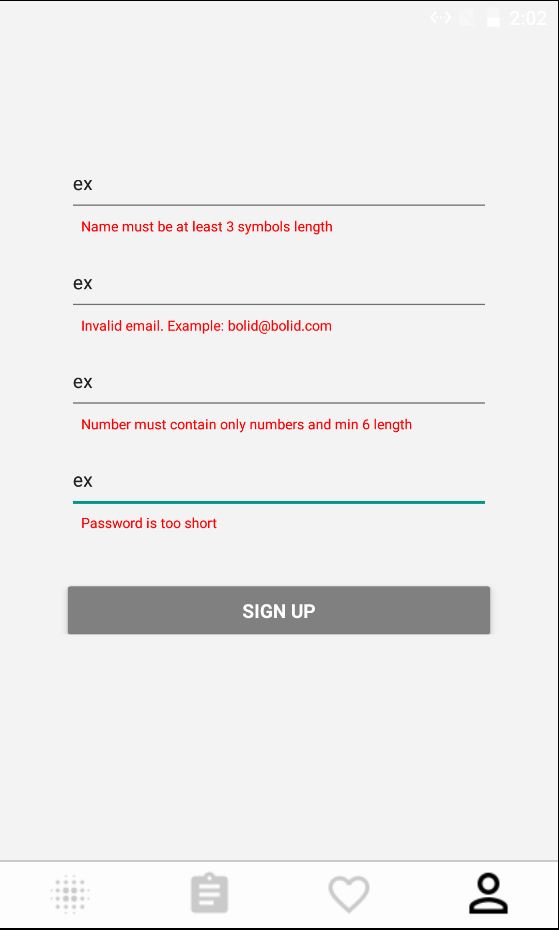


Рисунок 8.1 – Неверные данные

Также, необходимо проверить реакцию системы на подключение нескольких клиентов одновременно. Если к серверу подключается несколько клиентов, то в консоли выводится количество активных подключений. (Рисунок 8.2)

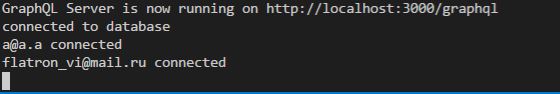


Рисунок 8.2 – Статистика активных клиентов сервера

# 

# 

# **ЗАКЛЮЧЕНИЕ**

Информационные технологии не стоят на месте, и порой невозможно проследить за выпуском нового программного обеспечения. Но именно это призвано значительно упростить жизнь человека, максимально автоматизируя всевозможные процессы.

Данное приложение создано для выполнения различных функции и запросов. Также оно сокращает время из-за автоматизации различных процессов и выявляет дополнительные ресурсы.

Также возможно расширение функций как у администратора, так и у пользователя. В будущем данный продукт будет совершенствоваться посредством добавления новых функций, улучшения уже существующих функций, добавления новых данных и внедрением новых возможностей.

Вся информация хранится в базе данных.

В программе предусмотрено использование следующих аспектов:

* исполняемые файлы работают в среде android 4.0 и выше
* интерфейс программы англоязычный.

Выполнено моделирование с использованием стандартов IDEF0 (функциональное моделирование процессов предметной области решаемой задачи (не менее чем 4 уровня)). Использовано CASE средство allfusion process modeler. Использовано CASE средство allfusion erwin data modeler. UML 2.0 (модели представления системы на основе UML).

Для описания работы приложения представлен следующий графический материал:

* диаграмма вариантов использования (Use Case);
* диаграмма состояний (Statechart);
* диаграмма последовательностей (Sequence diagram);
* диаграмма компонентов (component diagram);
* диаграмма развертывания (deployment diagram);
* диаграмма классов (Class diagram);
* информационная модель;
* блок-схемы алгоритмов, реализующих бизнес-логику
* IDEF0 – диаграмма операции выдачи книги.

Итогом курсового проекта является разработанное графическое приложение, которое позволит автоматизировать продажу одежды и облегчит этот процесс как для покупателей, так и для владельцев магазинов.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Дэвид Флэнаган JavaScript: Подробное руководство 1996 г – 324с.

[2] Дуглас Крокфорд JavaScript: сильные стороны. 2008 г – 192с

[3] Современный учебник Javascript [Электронный ресурс].

– Электронные данные. – Режим доступа: https://learn.javascript.ru/

[4] React Native [Электронный ресурс]. – Электронные данные.

– Режим доступа: <https://facebook.github.io/react-native/docs/style.html>

[5] Express – фреймворк веб-приложений Node.js. – Электронные данные. – Режим доступа: http://expressjs.com/ru/

[6] Introdution to GraphQL. – Электронные данные.

- Режим доступа: http://graphql.org/learn/

[7] Руководство по MongoDB. – Электронные данные. – Режим доступа: https://metanit.com/nosql/mongodb/

[8] Акила К, Еремеевский А. А.: Интернет-магазин с нуля. Полное пошаговое руководство. 2013г – 176с.

[9] Соловьев Д. С., Писарев А. А.: Интернет-магазин без правил. 2015г – 176с.

# **ПРИЛОЖЕНИЕ А**

**UML модели**

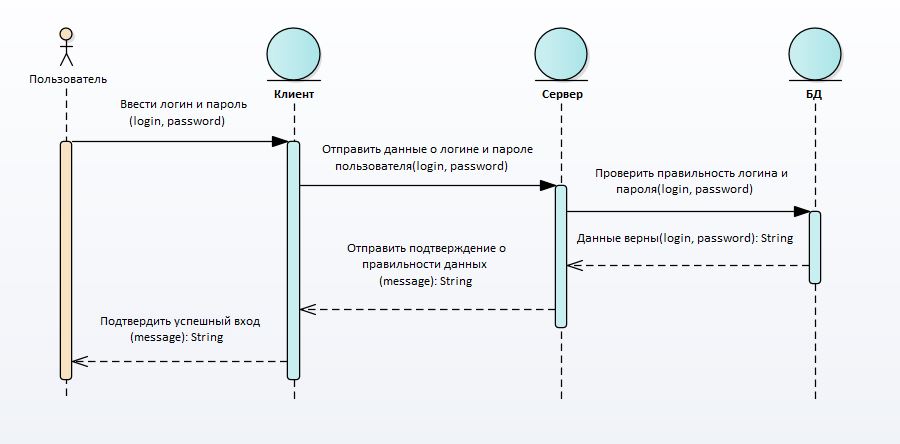
****

Рисунок А.1 – Диаграмма последовательности процесса авторизации

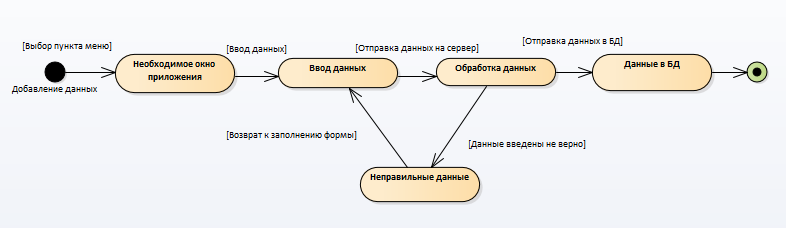
****

Рисунок А.2 Диаграмма состояний при добавлении товара

# **ПРИЛОЖЕНИЕ Б**

**Блок-схемы алгоритмов**

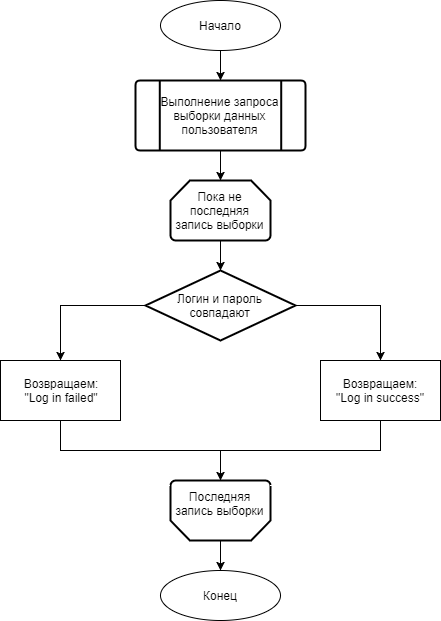


Рисунок Б.1 – Алгоритм авторизации

Продолжение приложения Б

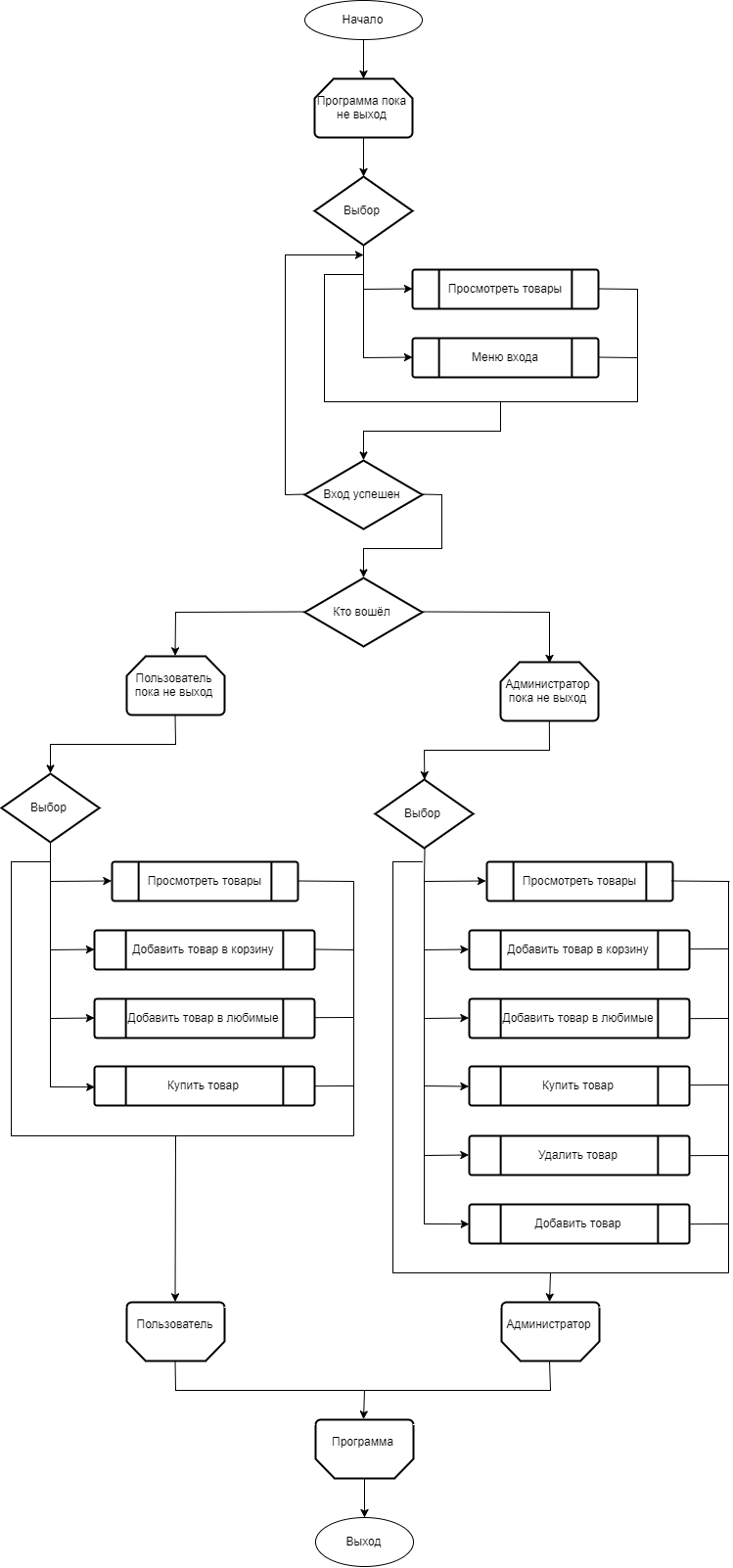


Рисунок Б.2 – Алгоритм работы всей программы

# **ПРИЛОЖЕНИЕ В**

**Листинг кода**

Файл app.js

const express = require('express');

const bodyParser = require('body-parser');

const Mongoose = require('mongoose');

const PORT = 3000;

const app = express();

const { graphiqlExpress } = require('apollo-server');

const { graphqlExpress } = require('apollo-server-express');

const { makeExecutableSchema } = require('graphql-tools');

Mongoose.Promise = global.Promise;

Mongoose.connect('mongodb://localhost:27017/users', (err) => {

if (err) {

return err;

}

console.log('connected to database');

return true;

});

const Schema = require('./schema');

const Resolvers = require('./resolvers');

const Connectors = require('./connectors');

const executableSchema = makeExecutableSchema({

typeDefs: Schema,

resolvers: Resolvers,

});

app.use('/graphql', bodyParser.json(), graphqlExpress(request => ({

schema: executableSchema,

context: {

token: request.headers.authorization,

constructor: Connectors,

},

})));

app.use('/graphiql', graphiqlExpress({

endpointURL: '/graphql',

}));

Продолжение приложения В

app.listen(PORT, () => console.log(

`GraphQL Server is now running on http://localhost:${PORT}/graphql`

));

Файл resolvers.js

const { createUserNote, createClothingNote } = require('./db/utils/uitls');

const resolveFunctions = {

Mutation: {

checkUser: async function checkUser(\_, { email, password }, ctx) {

let userData;

const user = new ctx.constructor.User();

await user.findUser(email, password)

.then((data) => {

userData = data;

})

.catch(e => console.log(e));

if (userData) {

console.log(`${userData.email} connected`);

return {

token: `${userData.username}\_${userData.username.length}`,

message: 'Log in success',

username: `${userData.username}`,

isAdmin: userData.isAdmin,

favourites: userData.favourites,

basket: userData.basket,

};

}

return { message: 'Log in failed' };

},

addUser: function addUser(root, data, ctx) {

console.log(ctx);

createUserNote(data);

const { email, password, username, mobile, isAdmin, favourites, basket } = data;

return { email, password, username, mobile, isAdmin, favourites, basket };

},

checkClothing: async function checkClothing(\_, { id }, ctx) {

let clothingData;

const clothing = new ctx.constructor.Clothing();

await clothing.findClothingById(id)

Продолжение приложения В

.then((data) => {

clothingData = data;

});

if (clothingData) {

console.log(clothingData);

return clothingData;

}

return { message: 'Getting clothing failed' };

},

addClothing: function addClothing(root, data, ctx) {

console.log(ctx);

createClothingNote(data);

const { name, price, filter, topic, images, sizes } = data;

return { name, price, filter, topic, images, sizes };

},

getClothings: async function getClothings(\_, $, ctx) {

let clothingData;

const clothing = new ctx.constructor.Clothing();

await clothing.findClothings()

.then((data) => { clothingData = data; });

if (clothingData) {

return clothingData;

}

return { message: 'Getting clothing failed' };

},

removeClothing: async function removeClothing(\_, { id }, ctx) {

const clothing = new ctx.constructor.Clothing();

await clothing.removeClothing(id)

.then(data => console.log(data));

return id;

},

updateFavourties: async function updateFavourties(\_, { email, item }, ctx) {

const user = new ctx.constructor.User();

await user.updateFavourties(email, item);

return item;

},

updateBasket: async function updateBasket(\_, { email, item }, ctx) {

const user = new ctx.constructor.User();

await user.updateBasket(email, item);

return item;

},

Продолжение приложения В

},

};

module.exports = resolveFunctions;

Файл schema.js

const typeDefinitions = `

type User {

email: String,

password: String,

username: String,

mobile: String,

isAdmin: Boolean,

favourites: [String],

basket: [String]

}

type Clothing {

\_id: String,

name: String,

price: Int,

filter: String,

topic: String,

images: [String],

sizes: [Int]

}

type Login {

token: String,

message: String,

username: String,

isAdmin: Boolean,

favourites: [String],

basket: [String]

}

type Mutation {

checkUser(email: String, password: String): Login

addUser(email: String, password: String, username: String, mobile: String): User

addClothing(name: String, price: Int, filter: String, topic: String, images: [String], sizes: [Int]): Clothing

checkClothing(id: String): Clothing

getClothings(something: String): [Clothing]

Продолжение приложения В

removeClothing(id: String): String

updateFavourites(email: String, item: Clothing): Clothing

updateBasket(email: String, item: Clothing): Clothing

}

type Query {

email: String

}

schema {

query: Query,

mutation: Mutation

}

`;

module.exports = [typeDefinitions];

# **ПРИЛОЖЕНИЕ Г**

**Листинг запросов**

Файл connectors.js

const { UserModel, ClothingModel } = require('./model');

const { find, without } = require('lodash');

class User {

constructor() {

this.findUser = (email, password) => {

const user = UserModel.findOne({ email, password }).exec();

return user;

};

this.updateFavourites = (email, item) => {

const user = UserModel.findOne({ email }).exec();

if (find(user.favourites, item)) {

user.favourites = without(user.favourites, item);

} else {

user.favourites.push(item);

}

user.save();

return user;

};

this.updateBasket = (email, item) => {

const user = UserModel.findOne({ email }).exec();

if (find(user.basket, item)) {

user.basket = without(user.basket, item);

} else {

user.basket.push(item);

}

user.save();

return user;

};

}

}

class Clothing {

constructor() {

this.findClothingById = (\_id) => {

const clothing = ClothingModel.findOne({ \_id }).exec();

return clothing;

};

this.findClothings = () => {

Продолжение приложения Г

const clothing = ClothingModel.find().exec();

return clothing;

};

this.removeClothing = (\_id) => {

console.log(\_id);

const clothing = ClothingModel.findOneAndRemove({ \_id }).exec();

return clothing;

};

}

}

module.exports = { User, Clothing };