Обложка

Разработка информационной системы автоматизации предоставления услуг банком

Задание

Аннотация

Пояснительная записка содержит 94 страницы. В ней имеются 42 рисунка, 13 таблиц, 16 источников.

Тема квалификационной работы посвящена разработке информационной системы автоматизации предоставления услуг банком.

Освящён процесс проектирования и создания базы данных, в которой хранится и обрабатывается всевозможная информация клиентах банка, покупках продуктов банка и самих продуктах.

Приведены пояснения по работе с информационной системой, присутствует руководство пользователя.

Annotation

This explanatory note contains 94 pages, 42 pictures, 13 tables, 16 sources.

The topic of the qualification work is devoted to the development of an information system for automating the provision of services by the bank.

The process of designing and creating a database in which all kinds of information is stored and processed by bank customers, purchases of bank products and the products themselves is consecrated.

There are explanations on the work with the information system, there is a user manual.

Оглавление

Введение

Одной из популярных форм оформления банковских продуктов становится – электронная форма. В качестве инструментальных средств реализации подобных задач используются web-приложения.

При этом под веб-приложением понимается приложение, взаимодействующее с сотрудниками банка, которые с любого устройства при помощи браузера открывают страницы и отправляют запросы к серверной части, а сервер обрабатывает полученные запросы, производит «общение» с базой данных и формирует новые web-страницы, которые отправляет обратно пользователю по протоколу HTTPS. Данный способ организации работы приложений называется технология «клиент-сервер».

Цель работы: разработать информационную систему с web-приложением для банковской деятельности, которая будет автоматизировать некоторые этапы продажи банковских продуктов.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ уже разработанных web сервисов для продажи банковских продуктов
- определить инструменты разработки web-приложения;
- провести анализ деятельности банка, в частности, по процессу оформления продуктов;
- сформировать требования к web-приложению;
- спроектировать и разработать web-приложение продажи банковских продуктов;
- разработать руководства пользователя для пользователей приложения;

Объектом работы является организация и продажа банковских продуктов.

Предмет работы: использование методов и средств автоматизации деятельности по организации и продажи банковских продуктов.

Квалификационная работа состоит из шести разделов.

В разделе 1 «Постановка задачи разработки» проводится первоначальное исследование потребностей пользователей, рассматриваются требования к разрабатываемой системе.

В разделе 2 «Технико-экономическое обоснование» обоснована актуальность темы работы, дана оценка новизны работы, обоснованы практическая ценность проекта и выбор средств разработки с указанием возможных альтернатив такого выбора.

В разделе 3 «Теоретическая часть» содержится описание функциональности и поведения информационной системы.

В разделе 4 «Практическая часть» содержится описание проектирования структуры базы данных для хранения информации.

В разделе 5 «Программная документация» содержатся описание применения программы, руководство оператора, руководство программиста.

В разделе 6 «Тестирование системы» содержится тестирование, полученной в ходе разработки, системы.

В приложении приведен прокомментированный листинг наиболее значимых частей программы.

1 ПОСТАНОВКА ЗАДАЧИ

Продажа банковских продуктов должно состоять из следующих этапов:

- Авторизация в системе;
- Формирование и редактирование продуктов;
- Продажа продуктов;
- Ведении истории удаленных продаж;
- Формирование отчетной документации по продажам.

Пользователи системы делятся на:

- GIDA – пользователи, которые могу вносить новых сотрудников и активировать/деактивировать учетные записи;
- team-leader – пользователи, которые могу заводить новые продукты, удалять (делает продукт неактивным, то есть он становится недоступным для продажи). Формировать отчеты по продажам, за выбранный период, по конкретному сотруднику или сразу по всем сотрудникам. Просматривать историю удаленных продаж, за выбранный период по конкретному сотруднику или сразу по всем сотрудникам;
- DSA – пользователи, которые могу оформлять продажи и удалять продажи за текущий день, для удаления доступны продажи пользователя, который зашел в систему. Формировать отчеты по продажам, за выбранный период, отчет содержит информацию, о пользователе, который зашел в систему;

Информационные потребности сотрудников с привилегиями GIDA:

- Авторизация по логину и паролю в системе;
- Просмотр главной станицы;
- Управление сотрудниками, добавление, изменение и удаление (делает пользователя неактивным);
- Просмотр страницы с контактами и формой обратной связи;

Информационные потребности сотрудников с привилегиями team-leader:

- Авторизация по логину и паролю в системе;

- Просмотр главной станицы;
- Управление продуктами, добавление, изменение и удаление (делает продукт неактивным, то есть он становится недоступным для продажи);
- Отчет по продажам, доступен выбор периода продажи, а также есть возможность вывода по конкурентному сотруднику, либо сразу по всем;
- История удаленных продаж, доступен выбор периода удаления, а также есть возможность вывода по конкурентному сотруднику, либо сразу по всем;
- Просмотр страницы с контактами и формой обратной связи;

Информационные потребности сотрудников с привилегиями DSA:

- Авторизация по логину и паролю в системе;
- Просмотр главной станицы;
- Управления продажами, оформление, и удаление продажи за текущий день, для удаления доступны продажи пользователя, который зашел в систему;
- Отчет по продажам, доступен выбор периода продажи, отчет выводится по пользователю, который зашел в системы;
- Просмотр страницы с контактами и формой обратной связи;

Web-приложение для продажи банковских продуктов должно включать следующие модули:

- Авторизация сотрудников, модуль доступен всем пользователям;
- Главная – информация о разработчике, модуль доступен пользователям с привилегиями: GIDA, team-leader и DSA;
- Сотрудники – управление сотрудниками, модуль доступен пользователям с привилегиями: GIDA;
- Продукты – управление продуктами, модуль доступен пользователям с привилегиями: team-leader;
- Отчет – отчет по продажам, модуль доступен пользователям с привилегиями: team-leader и DSA;

- История – история удаленных продаж, модуль доступен пользователям с привилегиями: team-leader;
- Контакты – контактная информация о разработчике, модуль доступен пользователям: GIDA, team-leader и DSA;
- Продажа продуктов – управление продажами, модуль доступен пользователям с привилегиями: DSA;

Web-приложение будет эффективно использоваться, если все пользователи системы будут выполнять инструкции в соответствии с руководством пользователя.

В качестве требования к производительности необходимо учитывать, что работа любого скрипта не должна превышать 15 секунд. При условии нагрузки на сервер не более 100 000 обращений к страницам web-приложения единовременно.

В качестве требования к надежности определяются возможностями системы прав доступа, заложенными разработчиком в системе и средствами администрирования серверной машины.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени на перезагрузку задействованных технических и программных средств при условии соблюдения условий эксплуатации самих технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановку программных средств.

Отказы веб-приложения возможны вследствие некорректных действий пользователя. Во избежание возникновения отказов программы по указанной

выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

Требования к исходным кодам и языкам программирования

- HTML 5
- CSS 3;
- SASS;
- JavaScript;
- jQuery;
- PHP;
- SQL

Требования к программным средствам, используемым веб-приложением:

- Операционная система Unix (RHEL, Debian и прочие);
- Веб-сервер Apache 2.4 и выше или Nginx;
- PHP 5.5 и выше (может быть собран как модуль Apache или как CGI-скрипт);
- СУБД MySQL 8 и выше;

Клиентская часть системы требует только установки веб-браузера на компьютере пользователя. Во всех современных операционных системах браузеры входят в стандартную комплектацию системы.

Требования к защите информации веб-приложения

Требования к защите определяются возможностями системы прав доступа, заложенными разработчиком в системе и средствами администрирования серверной машины.

Практическая значимость работы заключается в том, что разработка и внедрение в деятельность банка информационной системы с web-приложением, приведет к оптимизации некоторых этапов процесса продажи банковских продуктов, крайне благоприятно скажется на рабочем процессе.

2 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ТЕМЫ

2.1 Анализ предметной области

В настоящее время эффективное функционирование современного предприятия невозможно без применения информационных систем. Эта проблема актуальна как для крупных предприятий, так и для предприятий среднего и даже малого бизнеса. Информационные системы имеют ряд существенных отличий от стандартных прикладных программ. В зависимости от предметной области информационные системы могут сильно различаться по своей архитектуре и функциям.

Товаром на банковском рынке являются банковские продукты и услуги, оказываемые банком клиентам или другим банкам. Одной из популярных форм оформления банковских продуктов становится – электронная форма. В качестве инструментальных средств реализации подобны задач используются web-приложения.

Банковские продукты – отдельная банковская услуга или несколько связанных банковских услуг, которые банк предлагает клиентам на типичных условиях. Например, потребительский кредит, целевой вклад, депозитный сертификат, пластиковые карты и тому подобное.

Банковский продукт — это конкретный банковский документ (свидетельство), который производится банком для обслуживания клиента и проведения операции. Это может быть вексель, чек, банковский процент, депозит, любой сертификат и т.п.

Не стоит путать банковский продукт и банковскую услугу. Сходство банковского продукта и банковской услуги в том, что они призваны удовлетворить потребности клиента и способствовать получению прибыли. Так, банковский процент по депозитам является банковским продуктом, а его постоянная выплата представляет собой банковские услуги. И в том и в другом случае это служит удовлетворению потребности клиента и получению дохода. Открытие банковского счета — это продукт, а

обслуживание по счету — услуга, но при этом и продукт, и услуга предполагают получение дохода в виде комиссионных. В то же время следует подчеркнуть, что в большинстве случаев банковский продукт носит первичный характер, а банковская услуга — вторичный.

В отличие от продукта промышленного предприятия банковский продукт не выглядит зачастую как нечто материальное, вещественное. Кредиты и расчеты совершаются в форме записей по счетам, в безналичной денежной форме. Банковский продукт нельзя складировать, производить про запас.

Исходя из постоянно растущей популярности банковских продуктов и постоянно появляющихся новых предложениях, становится очевидно, что существует необходимость в автоматизации продаж продуктов банков, с целью ускорения работы соответствующих сотрудников, оптимизации хранения данных и удобного анализа имеющейся информации.

2.2 Анализ существующих разработок

В ходе выполнения исследования, к сожалению, не удалось подробно познакомится с системами других банков, так как такого рода системы являются внутренними и доступа у обычных пользователей нет.

Но можно посмотреть, как происходит оформление продуктов обычными пользователями у банка Тинькофф и Сбербанк.

На сайте Тинькофф необходимо заполнить онлайн формы на сайте, а затем с пользователем свяжется сотрудник (рисунок 2.1).

На сайте Сбербанка необходимо вначале войти Сбербанк Онлайн, а затем оформить продукт, либо вначале получить доступ к онлайн банкингу, а только затем оформить продукт (рисунок 2.2).

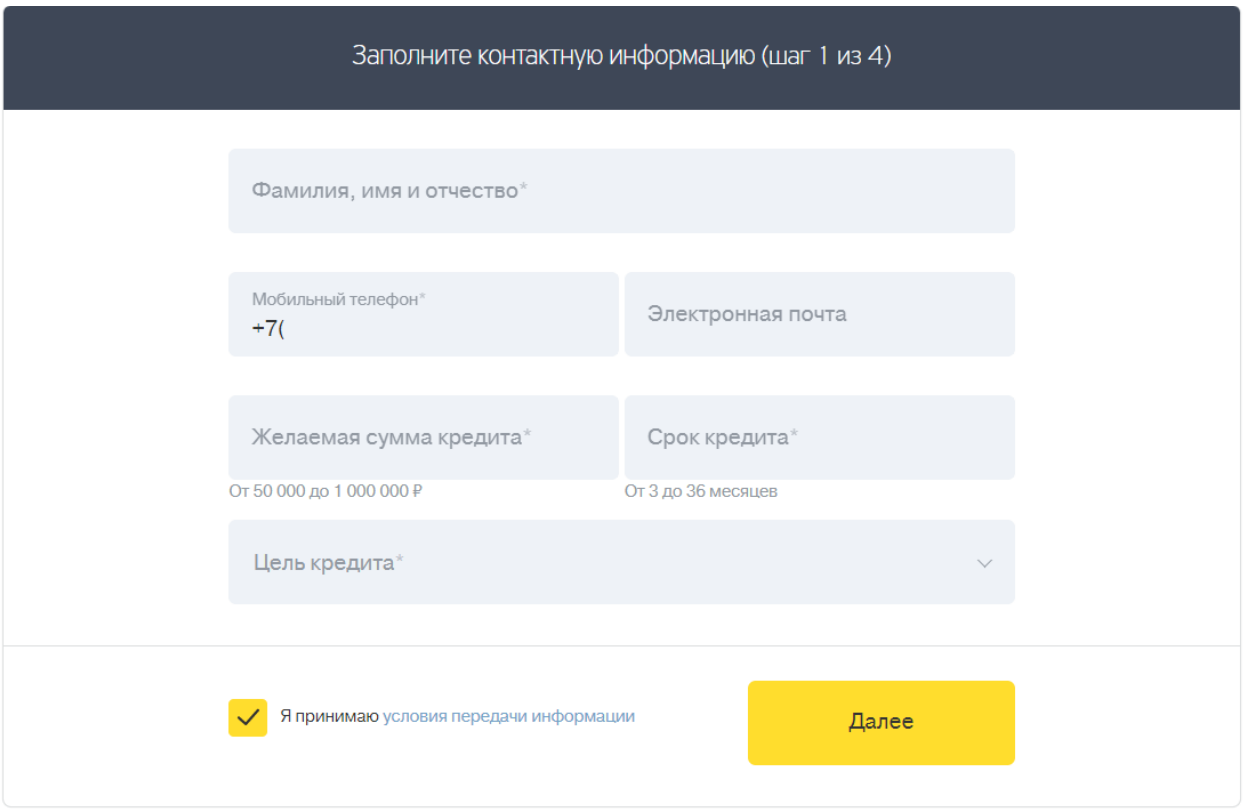

Рисунок 2.1 – Оформление продукта на сайте Тинькофф

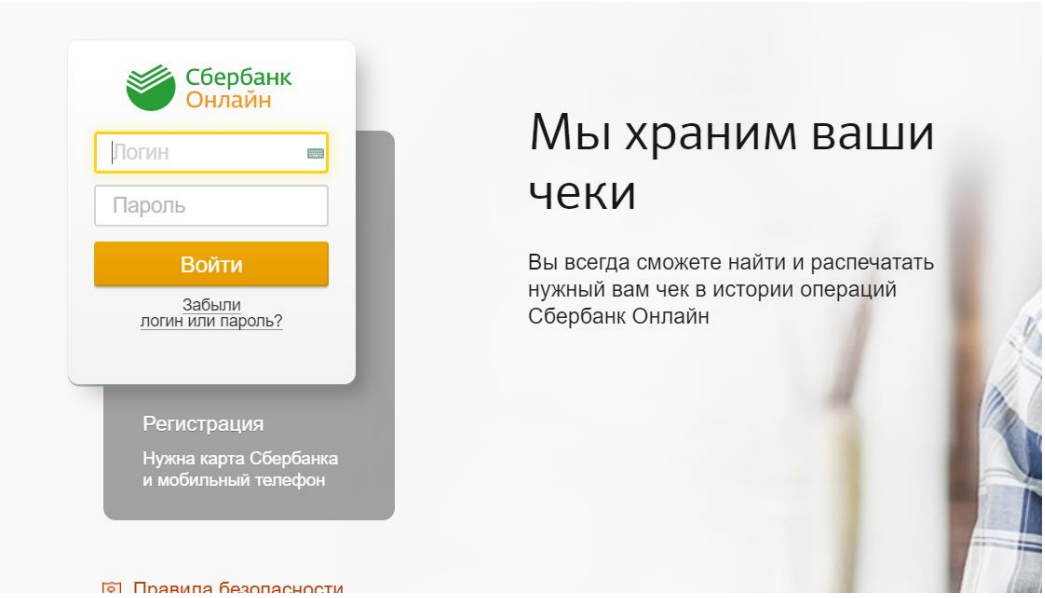

Рисунок 2.2 – Оформление продукта на сайте Сбербанка

Процесс оформления продукта очень похож в обоих банках, схематично он изображен на рисунке ниже.

Рисунок 2.3 – Этапы оформления продукта

2.3 Выбор средств моделирования и средств создания web-приложения

Для моделирования системы было выбрано облачное решение draw.io. Для разработки системы будут использоваться такие языки, как: JavaScript, PHP, системы управления базами данных MySQL, технология AJAX.

2.3.1 Draw.io

Draw.io – это инструмент для создания диаграмм и блок-схем онлайн. При этом огромное число шаблонов, которые позволяют нарисовать все что душе угодно. Инструментарий Draw.io очень напоминает MS Visio и возможно сделан под него, однако приложение от Microsoft программа платная, а онлайн сервис Draw.io – совершенно бесплатный, и главное не требует регистрации.

С помощью веб-сервиса Draw.io можно создавать:

- Диаграммы.
- UML-модели;
- Вставка в диаграмму изображений;
- Графики;

- Блок-схемы;
- Формы;
- Другое.

Для начала пользователь может выбрать объект из панели, просмотрев категории, и перенести мышью объект в документ. Для соединения объектов блок-схемы необходимо выделить второй объект и навести указателем на первый, далее появится зелёный флажок и с помощью него выполняется перетаскивание.

В меню сервиса диаграмму или схему можно отформатировать следующими настройками:

- Стиль шрифта;
- Цвет фона документа или объектов;
- Тени и степень прозрачности;
- Цвет и толщина линий;
- Заливка и градиент;

Также доступен экспорт готовых схем в изображение (PNG, GIF, JPG, PDF), синхронизация полученных документов с Google Диском и GitHub.

2.3.2 JavaScript

JavaScript – поддерживает объектно-ориентированный, императивный и функциональный стили. Является диалектом языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты

как списки, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

Вся уникальность данного языка программирования заключается в том, что он поддерживается практически всеми браузерами и полностью интегрируется с ними, а все что можно сделать с его помощью – делается очень просто. Ни одна другая технология не вмещает в себе все эти преимущества вместе. К примеру, есть такие, которые не кросс-браузерны (то есть поддерживаются не всеми браузерами). Это — VBScript, ActiveX, XUL. А есть такие, которые с браузером не интегрированы в нужной степени, это – Java, Flash, Silverlight. На сегодняшний день данная технология активно развивается, разрабатывается язык программирования Javascript 2.

2.3.3 PHP

В области программирования для сети Интернет PHP – один из популярных скриптовых языков (наряду с Perl и языками, используемыми в ASP.NET) благодаря богатой функциональности, простоте, скорости выполнения, кроссплатформенности и распространению исходных кодов на основе лицензии PHP.

Благодаря встроенным средствам для разработки веб-приложений, язык имеет большую популярность. Основными из них являются:

- Автоматическое извлечение POST и GET-параметров, и переменных окружения веб-сервера в предопределённые массивы;
- Взаимодействие с большим количеством различных систем управления базами данных;
- Автоматизированная отправка HTTP-заголовков;
- Работа с HTTP-авторизацией;
- Работа с cookies и сессиями;
- Работа с локальными и удалёнными файлами, сокетами;
- Обработка файлов, загружаемых на сервер.

Значительным отличием PHP от какого-либо кода, выполняющегося на стороне клиента, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. Вы даже можете сконфигурировать свой сервер таким образом, чтобы HTML-файлы обрабатывались процессором PHP, так что клиенты даже не смогут узнать, получают ли они обычный HTML-файл или результат выполнения скрипта.

PHP позволяет создавать качественные Web-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

PHP прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

2.3.4 PhpStorm

Для написания кода использовалась такой мощный инструмент как PhpStorm от компании JetBrains.

PhpStorm – коммерческая кросс-платформенная интегрированная среда разработки для PHP. Разрабатывается компанией JetBrains на основе платформы IntelliJ IDEA.

PhpStorm представляет собой интеллектуальный редактор для PHP, HTML и JavaScript с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами рефакторинга для PHP и JavaScript. Автодополнение кода в PhpStorm поддерживает спецификацию PHP 5.3, 5.4, 5.5, 5.6, 7.0 и 7.1 (современные и традиционные проекты), включая генераторы, сопрограммы, пространства имен, замыкания, типажи и синтаксис коротких массивов. Имеется полноценный SQL-редактор с возможностью редактирования полученных результатов запросов.

PhpStorm разработан на основе платформы IntelliJ IDEA, написанной на Java. Пользователи могут расширить функциональность среды разработки за счет установки плагинов, разработанных для платформы IntelliJ, или написав собственные плагины.

2.3.5 MySQL

MySQL является реляционной системой управления базами данных с открытым программным обеспечением.

MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

MySQL система, основанная на привилегиях и паролях, за счет чего обеспечивается гибкость и безопасность, и с возможностью верификации с удаленного компьютера. Пароли защищены, т.к. они при передаче по сети при соединении с сервером шифруются.

2.3.6 PhpMyAdmin

Для удобства работы с базами данных MySQL через браузер будет использоваться phpMyAdmin.

PhpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования системы управления базами данных MySQL. Данное приложение позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять системой управления базами данных MySQL без непосредственного ввода SQL команд, предоставляя дружественный интерфейс.

На сегодняшний день phpMyAdmin широко применяется на практике. Последнее связано с тем, что разработчики интенсивно развивают свой продукт, учитывая все нововведения системы управления базами данных MySQL. Подавляющее большинство российских провайдеров используют

это приложение в качестве панели управления для того, чтобы предоставить своим клиентам возможность администрирования выделенных им баз данных.

Приложение распространяется под лицензией GNU General Public License и поэтому многие другие разработчики интегрируют его в свои разработки, например, XAMPP, Denwer, AppServ, Open Server.

3 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

3.1 Рассмотрение подходов к разработке web-приложений

В наше время, время бурного роста интернет-технологий, существуют различные способы разработки сайтов. От мгновенных запусков в несколько часов, до продолжительных длящихся месяцами или даже годами. Рассмотрим наиболее популярные из них.

Самым простым способом, не требующим знаний языков программирования это использование визуальных редакторов, где путем преставления блоков заполняется вся структура и получается простейшие странички сайта. Так же есть огромное множество сервисов по созданию блогов, в которых сайт собирается как конструктор. Самыми распространёнными из них являются uCoz, Wix, uKit и т.д.

Если есть знания языка гипертекстовой разметки HTML, а также CSS (каскадные таблицы стилей), то можно сделать простой статический сайт. С помощью HTML строится разметка содержимого будущего сайта, а через CSS настраивается его внешний вид. Для создания более сложных динамических сайтов потребуется знания скриптового языка программирования PHP. Что бы сайт обладал большей функциональностью потребуются знания JavaScript и различных технологий, позволяющих сделать сайт более красивым, динамичным и быстродействующими. Такое создание с нуля займет уйму времени, как показывает практика, возникает множество ошибок и работы. Для устранения этих ошибок существуют такое программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта, называемое framework. Это каркас программной системы (или подсистемы), который может включать вспомогательные программы, библиотеки кода, язык сценариев и другое ПО, облегчающее разработку и объединение разных компонентов большого программного проекта. Обычно объединение происходит за счёт использования единого API. Framework определяется как множество

конкретных и абстрактных классов, а также определений способов их взаимоотношения. Конкретные классы обычно реализуют взаимные отношения между классами. Абстрактные классы представляют собой точки расширения, в которых каркасы могут быть использованы или адаптированы.

Еще одним более простым способом создания сайта является использование системы для обеспечения и организации совместного процесса создания, редактирования и управления содержимым страницы (CMS). В основном CMS имеют модульную архитектуру, и управление контентом происходит благодаря отдельным модулям. В большинстве CMS встроено множество готовых моделей, а также есть уже готовый шаблонный сайт, который можно начать редактировать под себя.

В данный момент существует большое множество CMS, как платных, так и бесплатных. CMS позволяют решать задачи различного уровня сложности в гораздо меньшие сроки, чем написание всего кода с нуля.

Но так, как для банка безопасность PI данных, гораздо дороже потраченного времени на разработку, то воспользуемся вторым подходом. Будем писать систему с нуля с использованием одобренных в компании framework.

3.2 Проектирование web-приложения информационной системы

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе. Диаграмма вариантов использования для web-приложения информационной системы изображена на рисунках ниже.

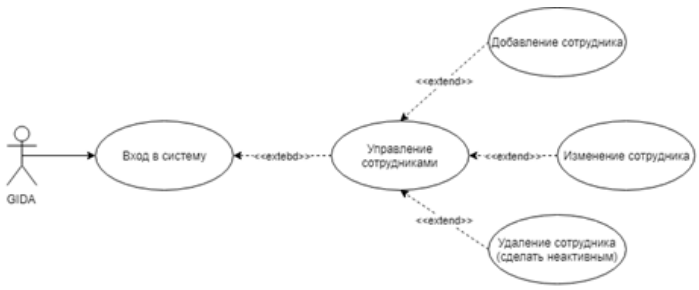

Рисунок 3.1 – UML диаграмма прецедентов для GIDA

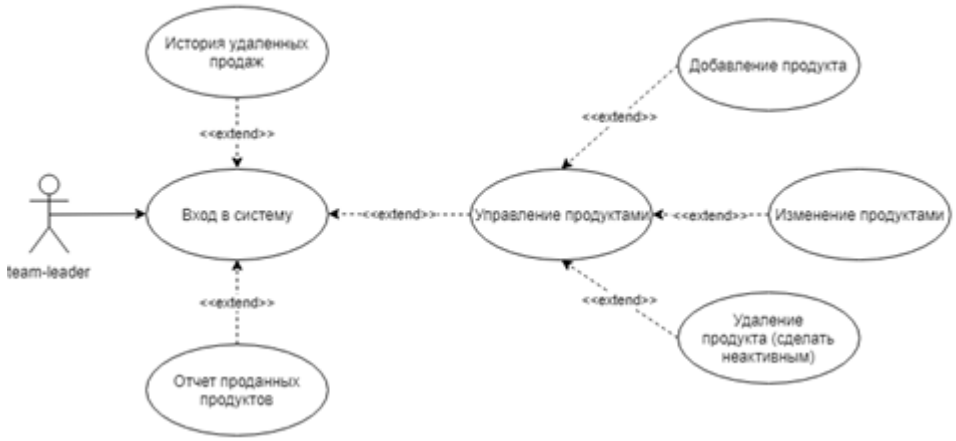

Рисунок 3.2 – UML диаграмма прецедентов для team leader

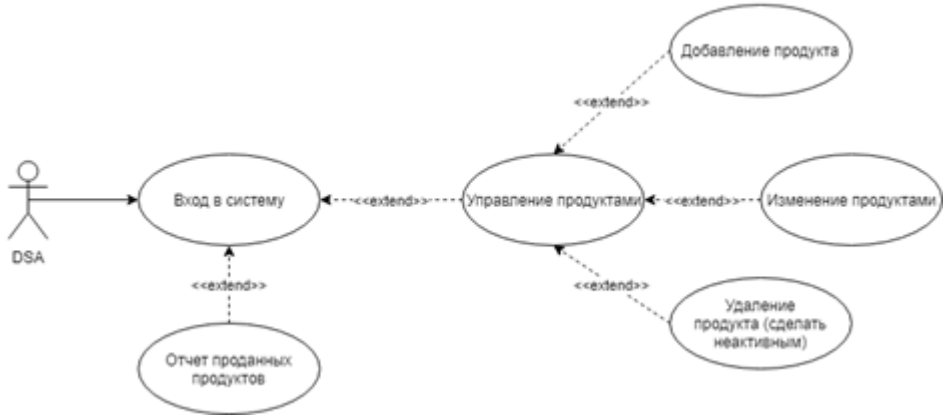

Рисунок 3.3 – UML диаграмма прецедентов для DSA

4 ПРАКТИЧЕСКАЯ ЧАСТЬ

4.1 Концептуальное проектирование

На этапе построения концептуальной модели необходимо определить цель создания БД, основные функции и информацию, которую БД должна содержать. Таким образом, нужно определить семантику будущих таблиц, а также характер содержащейся в них информации.

Проектируемая база данных должна соответствовать предметной области и обеспечивать все потребности в хранении информации, которую задействует предметная область.

Одним из наиболее сложных этапов проектирования, является разработка ER-диаграммы. Поскольку этот этап определяет вид базы данных, то ошибки, допущенные на этой стадии разработки, будут критичными в дальнейшем. Поэтому необходимо очень тщательно подходить к данному этапу.

Проанализировав предметную область, а также потребности предполагаемых пользователей системы были выявлены следующие сущности:

- клиент;
- город;
- покупка;
- покупка продуктов;
- платежная система;
- продукт;
- тип карты;
- сотрудник;
- роли;
- роли для страницы;
- страницы.

Рассмотрим сущности и связи между ними:

Связь «Покупка продуктов» – «Продукт» имеет степень связи 1..N, потому что каждый продукт может быть куплен несколько раз, а в каждой покупке он может относиться только единожды. Принадлежность у сущности «Покупка продуктов» обязательная, у сущности «Продукт» необязательная (рисунок 4.1);

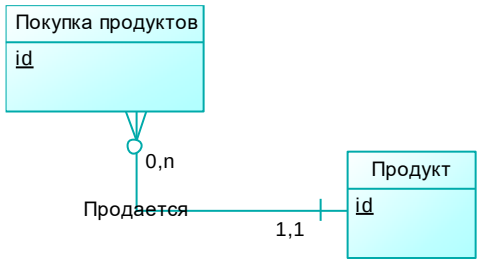

Рисунок 4.1 – Связь «Покупка продуктов» – «Продукт»

Связь «Платежная система» – «Продукт» имеет степень связи 1..N, потому что каждая платежная система может быть отнесена к нескольким продуктам, а каждый продукт может относиться только к одной системе. Принадлежность у сущности «Продукт» обязательная, у сущности «Платежная система» необязательная (рисунок 4.2);

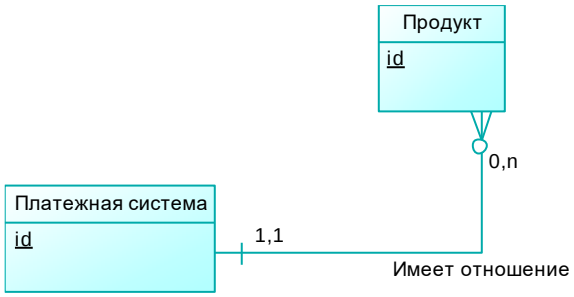

Рисунок 4.2 – Связь «Платежная система» – «Продукт»

Связь «Тип карты» – «Продукт» имеет степень связи 1..N, потому что каждый тип карты может привязать к себе много продуктов, а каждый продукт может относиться только к одному типу карты. Принадлежность у

сущности «Продукт» обязательная, у сущности «Тип карты» необязательная (рисунок 4.3).

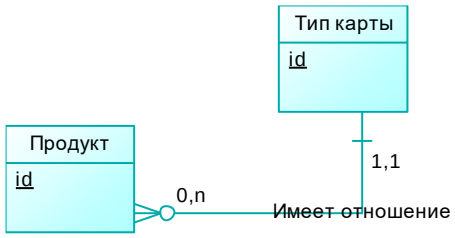

Рисунок 4.3 – Связь «Тип карты» – «Продукт»

Связь «Покупка» – «Покупка продукта» имеет степень связи 1..N, потому что каждая покупка продукта может содержать только конкретный продукт, а каждый продукт может быть продан множество раз. Принадлежность у сущности «Покупка продукта» обязательная, у сущности «Покупка» необязательная (рисунок 4.4).

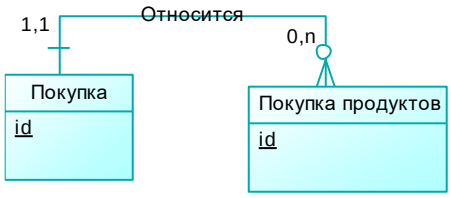

Рисунок 4.4 – Связь «Покупка» – «Покупка продукта»

Связь «Клиент» – «Покупка» имеет степень связи 1..N, потому что покупка относится к конкретному клиенту, а каждый клиент может сделать много покупок. Принадлежность у сущности «Покупка» обязательная, у сущности «Клиент» необязательная (рисунок 4.5).

Рисунок 4.5 – Связь «Клиент» – «Покупка»

Связь «Клиент» – «Город» имеет степень связи 1..N, потому что клиент относится к конкретному городу, а в каждом городе может жить много клиентов. Принадлежность у сущности «Клиент» обязательная, у сущности «Город» необязательная (рисунок 4.6).

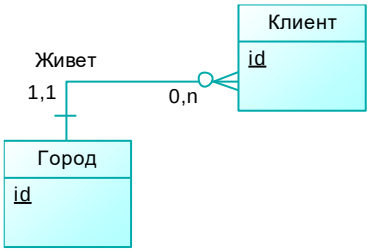

Рисунок 4.6 – Связь «Клиент» – «Город»

Связь «Покупка» – «Сотрудник» имеет степень связи 1..N, потому что покупку проводит конкретный сотрудник, а каждый сотрудник может провести много покупок. Принадлежность у сущности «Покупка» обязательная, у сущности «Сотрудник» необязательная (рисунок 4.7).

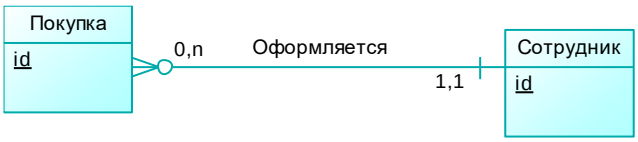

Рисунок 4.7 – Связь «Покупка» – «Сотрудник»

Связь «Роли» – «Сотрудник» имеет степень связи 1..N, потому что каждый сотрудник имеет конкретную роль, а каждая роль может относиться ко многим сотрудникам. Принадлежность у сущности «Сотрудник» обязательная, у сущности «Роли» необязательная (рисунок 4.8).

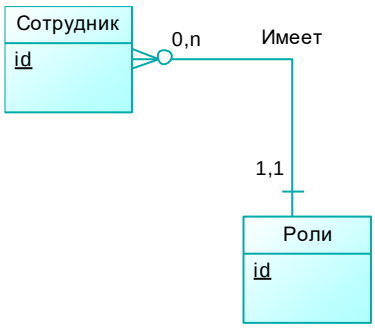

Рисунок 4.8 – Связь «Роли» – «Сотрудник»

Связь «Роли» – «Роли для страниц» имеет степень связи 1..N, потому что каждая роль может относиться к разным страницам, а каждая страница может относиться только к одной роли. Принадлежность у сущности «Роли для страниц» обязательная, у сущности «Роли» необязательная (рисунок 4.9).

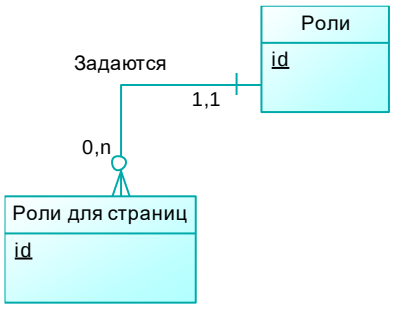

Рисунок 4.9 – Связь «Роли» – «Роли для страниц»

Связь «Страницы» – «Роли для страниц» имеет степень связи 1..N, потому что каждая страница имеет конкретную роль, а каждая роль может относиться ко многим страницам. Принадлежность у сущности «Роли для страниц» обязательная, у сущности «Страницы» необязательная (рисунок 4.10).

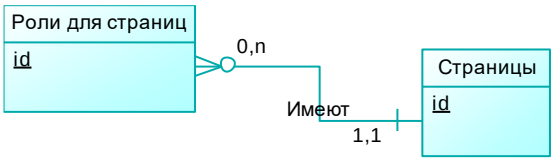

Рисунок 4.10 – Связь «Страницы» – «Роли для страниц»

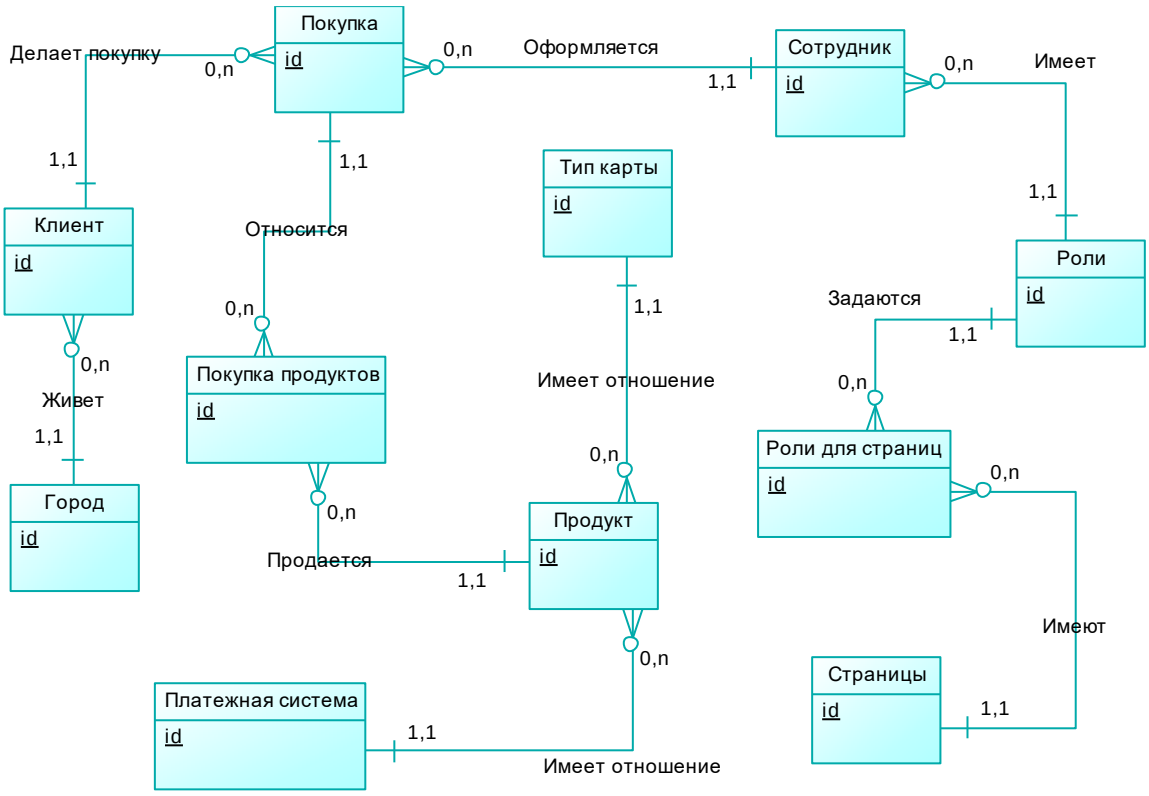

Рисунок 4.11 – Схема ER–диаграммы

4.2 Логическое проектирование

Для того чтобы перейти от построенной ER-диаграммы к предварительным отношениям необходимо использовать правила построения связей отношений.

Всего на практике применяют 7 правил, из которых первые 6 применяют для бинарных связей, а 7-е для тернарных связей и связей более высоких порядков. Поскольку в проектируемой базе данных не содержится тернарных связей и связей более высоких порядков, то рассмотрим только первые 6 правил.

Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей является обязательным, то требуется только одно отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

Если степень бинарной связи 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение двух отношений. Под каждую сущность выделяется одно отношение, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которого класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности.

Если степень бинарной связи равна 1:1 и класс принадлежности ни одной из сущностей не является обязательным, то необходимо использовать три отношения: по одному для каждой сущности и одно отношение для связи. Причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

Если степень бинарной связи равна 1:N и класс принадлежности N-связной сущности является обязательным, то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ 1-связной сущности должен быть добавлен как атрибут в отношение, отводимое N-связной сущности.

Если степень бинарной связи равна 1:N и класс принадлежности N-связной сущности является необязательным, то необходимо формирование 3-х отношений: по одному для каждой сущности и одно отношение для связи. Причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

Если степень бинарной связи равна M:N, то для хранения данных необходимо три отношения: по одному для каждой сущности и одно

отношение для связи. Причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

Далее необходимо в соответствии с вышеописанными правилами определить предварительные отношения между таблицами. (Полная логическая модель базы данных представлена на рисунке 4.23)

Выберем правило №4 для сущностей «Покупка» и «Клиент» (рисунок 4.12). Получим следующие предварительные отношения:

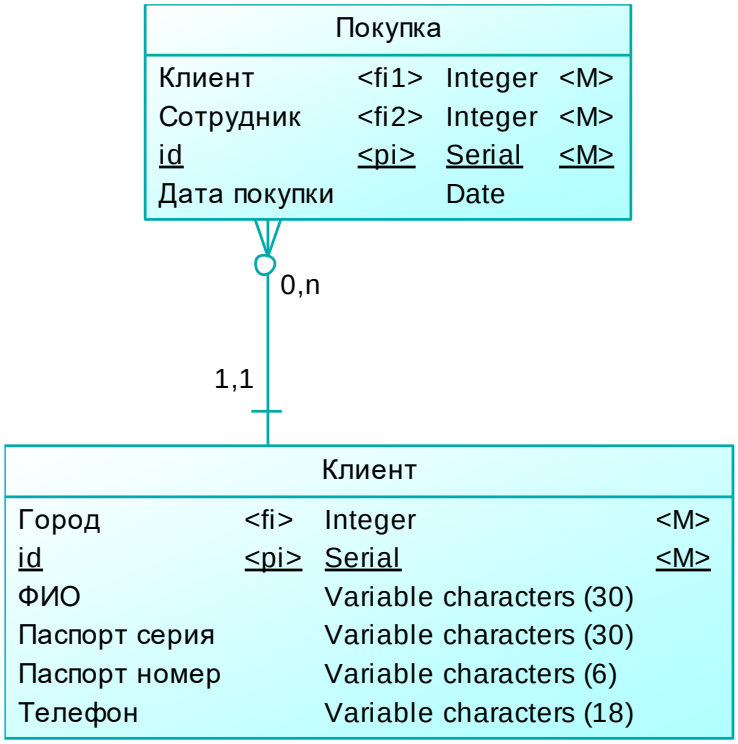

Рисунок 4.12 – Отношение
«Покупка» – «Клиент»

Выберем правило №4 для сущностей «Клиент» и «Город» (рисунок 4.13). Получим следующие предварительные отношения:

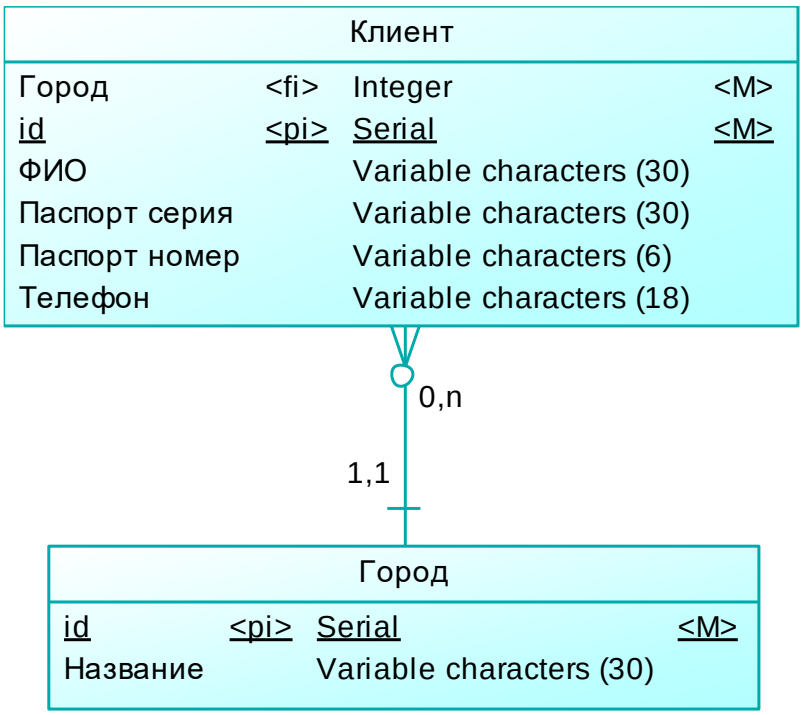

Рисунок 4.13 – Отношение «Клиент» – «Город»

Выберем правило №4 для сущностей «Покупка» и «Покупка продуктов» (рисунок 4.14). Получим следующие предварительные отношения:

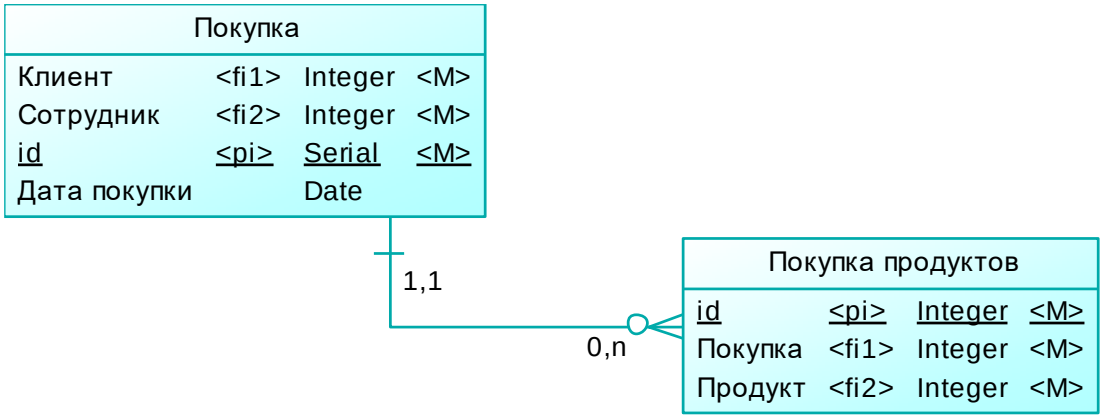

Рисунок 4.14 – Отношение «Покупка» – «Покупка продуктов»

Выберем правило №4 для сущностей «Покупка» и «Сотрудник» (рисунок 4.15). Получим следующие предварительные отношения:

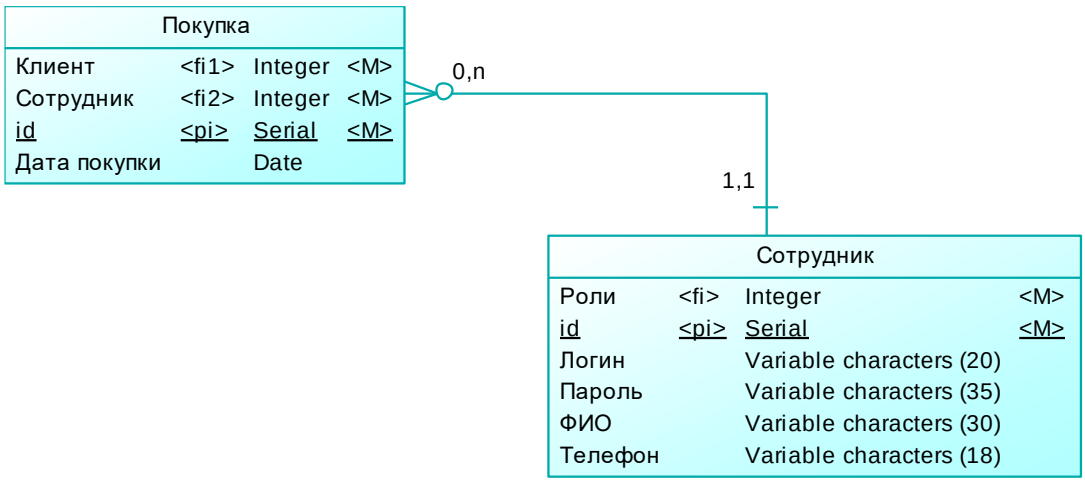

Рисунок 4.15 – Отношение «Покупка» – «Сотрудник»

Выберем правило №4 для сущностей «Сотрудник» и «Роли» (рисунок 4.16). Получим следующие предварительные отношения:

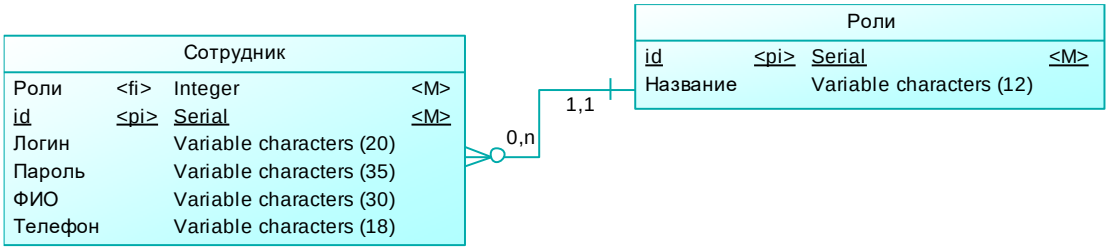

Рисунок 4.16 – Отношение «Сотрудник» – «Роли»

Выберем правило №4 для сущностей «Роли» и «Роли для страниц» (рисунок 4.17). Получим следующие предварительные отношения:

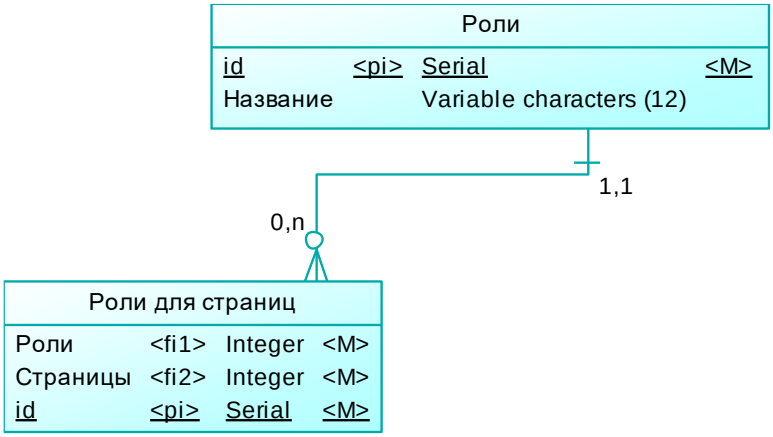

Рисунок 4.17 – Отношение «Роли» – «Роли для страниц»

Выберем правило №4 для сущностей «Страницы» и «Роли для страницы» (рисунок 4.18). Получим следующие предварительные отношения:

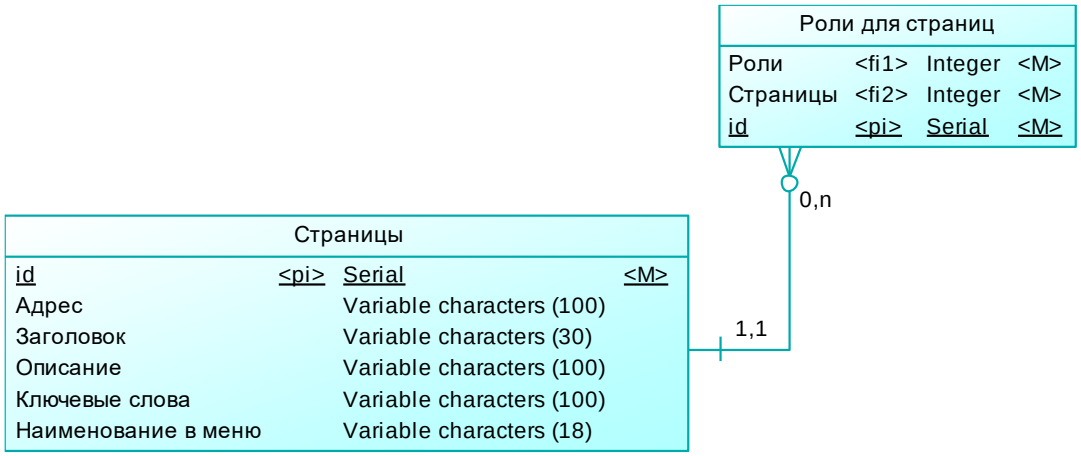

Рисунок 4.18 – Отношение «Страницы» – «Роли для страниц»

Выберем правило №4 для сущностей «Продукт» и «Тип карты» (рисунок 4.19). Получим следующие предварительные отношения:

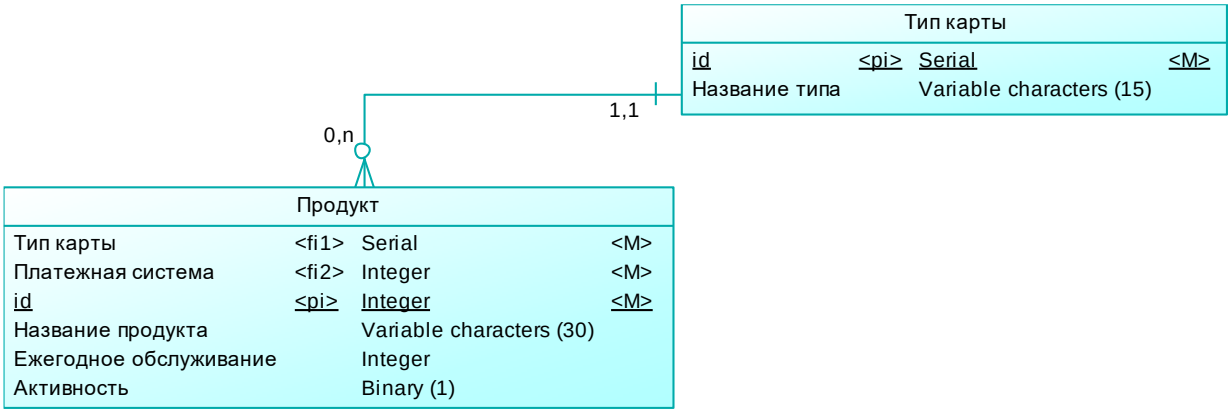

Рисунок 4.19 – Отношение «Продукт» – «тип карты»

Выберем правило №4 для сущностей «Продукт» и «Платежная система» (рисунок 4.20). Получим следующие предварительные отношения:

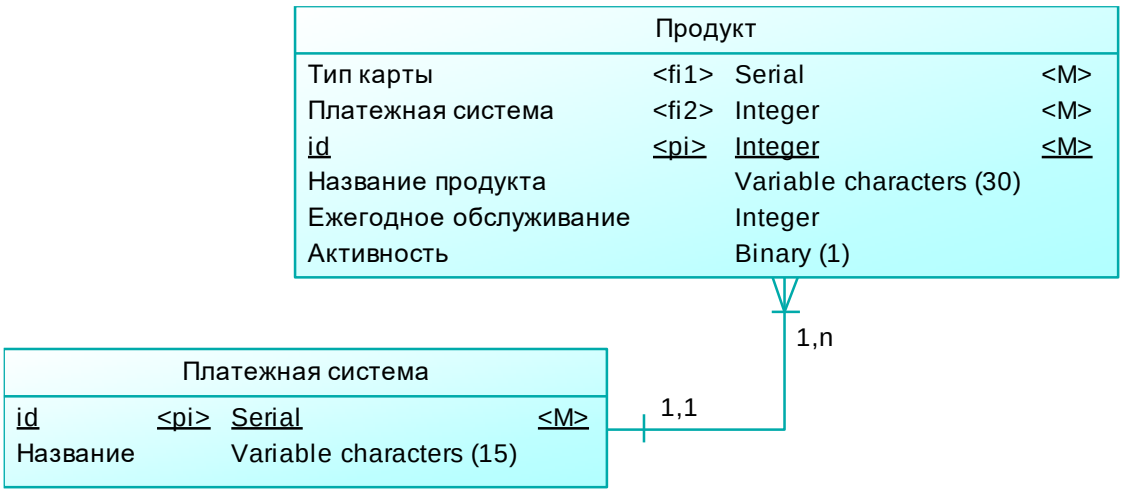

Рисунок 4.20 – Отношение «Продукт» – «Платежная система»

Выберем правило №4 для сущностей «Покупка продуктов» и «Продукт» (рисунок 4.21). Получим следующие предварительные отношения:

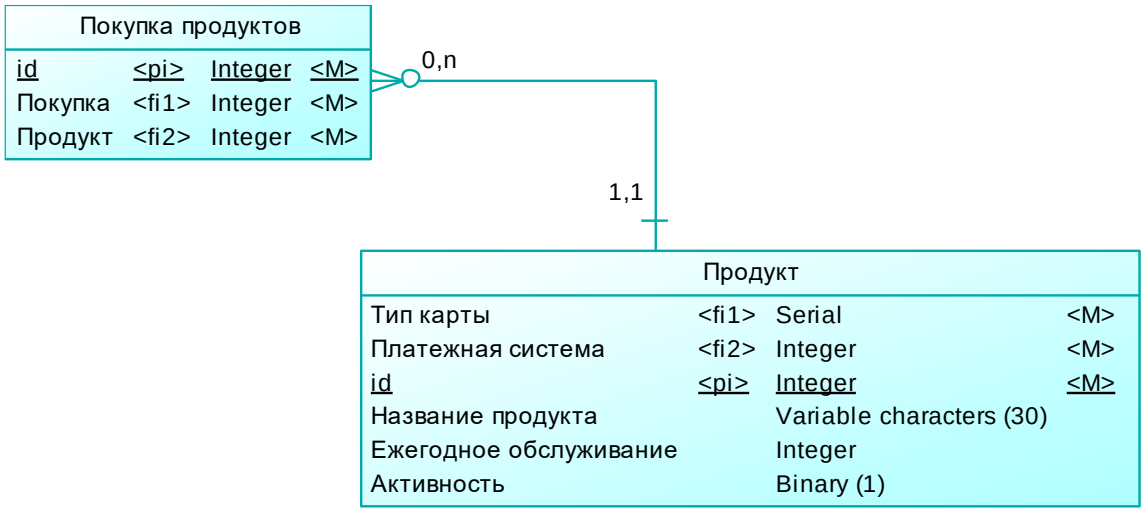

Рисунок 4.21 – Отношение «Продукт» – «Покупка продуктов»

Выберем правило №4 для сущностей «Покупка» и «Покупка продуктов» (рисунок 4.22). Получим следующие предварительные отношения:

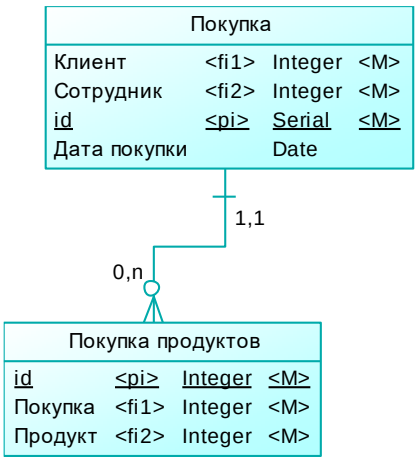

Рисунок 4.22 – Отношение «Покупка» – «Покупка продуктов»

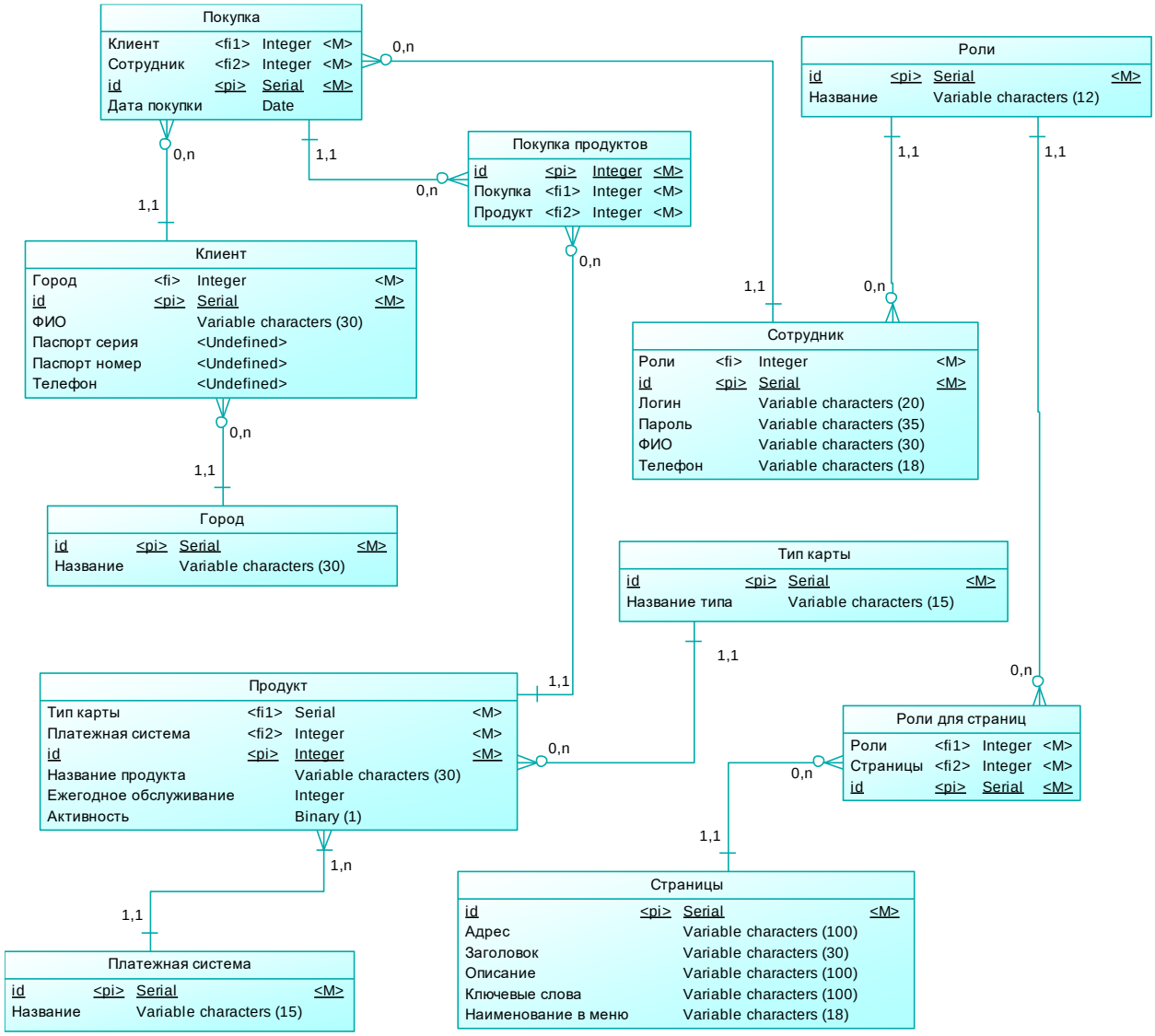

Рисунок 4.23 – Логическая модель базы данных

4.3 Разработка физической модели

Свойства таблиц представлены в таблицах ниже. Все эти таблицы имеют одинаковую структуру. Их назначение представлено ниже:

Поле – наименование таблицы;

Тип – тип данных, хранящихся в поле;

Индекс – это важный и полезный инструмент, который позволяет оптимизировать выборку из базы данных, значительно сокращая время на получение нужных данных. При этом заметить разницу можно на очень больших таблицах, содержащих десятки и сотни тысяч строк.

Первичный ключ (PRIMARY KEY) – это основной ключ, который в таблице может быть только один. Он позволяет идентифицировать уникальные записи в таблице. Значения, которые находятся в столбце, где поля имеют PRIMARY KEY, не могут повторяться. Нередко первичный ключ назначают для полей с идентификатором id.

Уникальный ключ (UNIQUE) – по сути, это альтернатива первичному ключу: значения, которые содержатся в таких полях также не могут повторяться и иметь значение NULL.

Индекс (INDEX) – обычный индекс (как мы описали выше). В Mysql, кроме того, можно индексировать строковые поля по заданному числу символов от начала строки.

a_i – признак авто заполнения поля, символ "*" обозначает, что поле заполняется автоматически путем прибавления единицы к предыдущему полю;

null – признак разрешение хранения значений типа NULL (символ «*» обозначает, что в поле разрешено хранение значений типа NULL).

city – таблица городов. В ней хранится список городов. Таблица связана с таблицей **client** по типу связи «один ко многим», т.е. для одного города может быть определено несколько клиентов. Свойства указаны в таблице ниже.

Таблица 4.1 – Список городов

Поле	Тип	Ключ	a_i	Описание
id	int(11)	PK	*	Уникальный номер города
name	varchar(30)	UNIQUE		Наименование города

client – таблица клиентов. В ней хранится список клиентов. Ссылка на таблицу **city** city.id ON UPDATE RESTRICT ON DELETE RESTRICT. Свойства указаны в таблице ниже.

Таблица 4.2 – Список клиентов

Поле	Тип	Ключ	a_i	Описание
id	int(11)	PK	*	Уникальный номер клиента
FLMName	varchar(30)			ФИО клиента
seriesPaspor	char(4)			Серия паспорта
idPasport	char(6)			Номер паспорта
phone	char(18)			Телефон
idCity	int(11)	INDEX		Уникальный номер города

employee – таблица сотрудников. В ней хранится список клиентов. Ссылка на таблицу **role** -> role.id ON UPDATE RESTRICT ON DELETE RESTRICT. Свойства указаны в таблице ниже.

Таблица 4.3 – Список сотрудников

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
id	int(11)	PK	*	Уникальный номер сотрудника
login	varchar(20)	UNIQUE		Логин сотрудника
password	char(32)			Пароль сотрудника
idRole	int(11)	INDEX		Уникальный номер роли
FLMName	varchar(30)			ФИО сотрудника
phone	char(18)			Телефон
active	tinyint(1)			Активна/неактивна учетная запись

historyPurchased– таблица истории проданных продуктов. В ней хранятся удаленные продажи продуктов. Ссылка на таблицы **client** -> client.id ON UPDATE RESTRICT ON DELETE RESTRICT **employee** -> employee.id ON UPDATE RESTRICT ON DELETE RESTRICT. Свойства указаны в таблице ниже.

Таблица 4.4 – История проданных продуктов

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
id	int(11)			Номер удаленной продажи
idClient	int(11)	INDEX		Уникальный номер клиента
idEmployee	int(11)	INDEX		Уникальный номер сотрудника
datePurchased	date			Дата покупки
dateRemoved	date			Дата удаления

historyPurchasedProduct – это связка между удаленными продажами (**historyPurchased**) и продуктами (**product**). Используется для составления соответствия между продажами и продуктами, связь между ними «многие-ко-многим». Свойства указаны в таблице ниже.

Таблица 4.5 – Связь продаж и продуктов

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
idPurchased	int(11)	INDEX		Номер удаленной продажи
idProduct	int(11)	INDEX		Уникальный номер продукта

pages – таблица страниц. В ней хранится список страниц. Таблица связана с таблицей **role** по типу связи «многие-ко-многим», т.е. для нескольких страниц может быть определено несколько ролей. Свойства указаны в таблице ниже.

Таблица 4.6 – Список страниц

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
id	int(11)	PK	*	Уникальный номер страницы
url	varchar(100)	UNIQUE		URL адрес
title	varchar(30)			Наименование
description	varchar(100)			Описание
keywords	varchar(100			Ключевые слова
titleMenu	varchar(18)			Наименование в меню

pagesRole – это связка между страницами (**pages**) и ролями (**role**). Используется для составления соответствия между страницами и ролями, т.е.

какие роли имеют доступ к страницам связь между ними «многие-ко-многим». Ссылка на таблицы **pages** -> pages.id ON UPDATE CASCADE ON DELETE RESTRICT и **role** -> role.id ON UPDATE RESTRICT ON DELETE RESTRICT. Свойства указаны в таблице ниже.

Таблица 4.7 – Связь страниц и ролей

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
idPages	int(11)	INDEX		Уникальный номер страницы
idRole	varchar(100)	INDEX		Уникальный номер роли

paymentSystem – таблица платежных систем. В ней хранится список платежных систем. Таблица связана с таблицей **product** по типу связи «один ко многим», т.е. для одной платежной системы может быть определено несколько продуктов. Свойства указаны в таблице ниже.

Таблица 4.8 – Список платежных систем

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
id	int(11)	PK	*	Уникальный номер страницы
name	varchar(15)	UNIQUE		Наименование платежной системы

product – таблица продуктов. В ней хранится список продуктов. Таблица связана с таблицей **purchased** по типу связи «многие-ко-многим», т.е. для нескольких продуктов может быть определено несколько продаж. Ссылка на таблицы **typeCard** -> typeCard.id ON UPDATE RESTRICT ON

DELETE RESTRICT и **paymentSystem** -> paymentSystem.id ON UPDATE RESTRICT ON DELETE RESTRICT Свойства указаны в таблице ниже.

Таблица 4.9 – Список продуктов

Поле	**Тип**	**Ключ**	**a_i**	**Описание**
id	int(11)	PK	*	Уникальный номер продукта
name	varchar (30)	UNIQUE		Наименование продукта
idCard	int(11)	INDEX		Уникальный номер типа карты
idPayment	int(11)	INDEX		Уникальный номер платёжной системы
annualMaintenance	smallint(6)			Ежегодное обслуживание
active	tinyint(1)			Активен/неактивен продукт

purchased – таблица проданных продуктов. В ней хранятся проданные продукты. Таблица связана с таблицей **product** по типу связи «многие-ко-многим», т.е. для нескольких продаж может быть определено несколько продуктов. Ссылка на таблицы **client** -> client.id ON UPDATE RESTRICT ON DELETE RESTRICT **employee** -> employee.id ON UPDATE RESTRICT ON DELETE RESTRICT. Свойства указаны в таблице ниже.

Таблица 4.10 – Проданные продукты

Поле	Тип	Ключ	a_i	Описание
id	int(11)	PK	*	Уникальный номер продажи
idClient	int(11)	INDEX		Уникальный номер клиента
idEmployee	int(11)	INDEX		Уникальный номер сотрудника
datePurchased	date			Дата покупки

purchasedProduct – это связка между продажами (**purchased**) и продуктами (**product**). Используется для составления соответствия между продажами и продуктами, связь между ними «многие-ко-многим». Свойства указаны в таблице ниже.

Таблица 4.11 – Связь продаж и продуктов

Поле	Тип	Ключ	a_i	Описание
idPurchased	int(11)	INDEX		Уникальный номер продажи
idProduct	int(11)	INDEX		Уникальный номер продукта

role – таблица ролей. В ней хранится список ролей. Таблица связана с таблицей **role** по типу связи «многие-ко-многим», т.е. для нескольких ролей может быть определено несколько страниц. Свойства указаны в таблице ниже.

Таблица 4.12 – Список ролей

Поле	Тип	Ключ	a_i	Описание
id	int(11)	PK	*	Уникальный номер ролей
name	varchar(15)	UNIQUE		Наименование ролей

typeCard – таблица типа карт. В ней хранится список типа карт. Таблица связана с таблицей **product** по типу связи «один ко многим», т.е. для одного типа карт может быть определено несколько продуктов. Свойства указаны в таблице ниже.

Таблица 4.13 – Список типа карт

Поле	Тип	Ключ	a_i	Описание
id	int(11)	PK	*	Уникальный номер типа карт
type	varchar(15)	UNIQUE		Наименование типа карт

Физическая модель данных представлена ниже (рисунок 4.24).

Схема развернутой базы данных представлена ниже (рисунок 4.25).

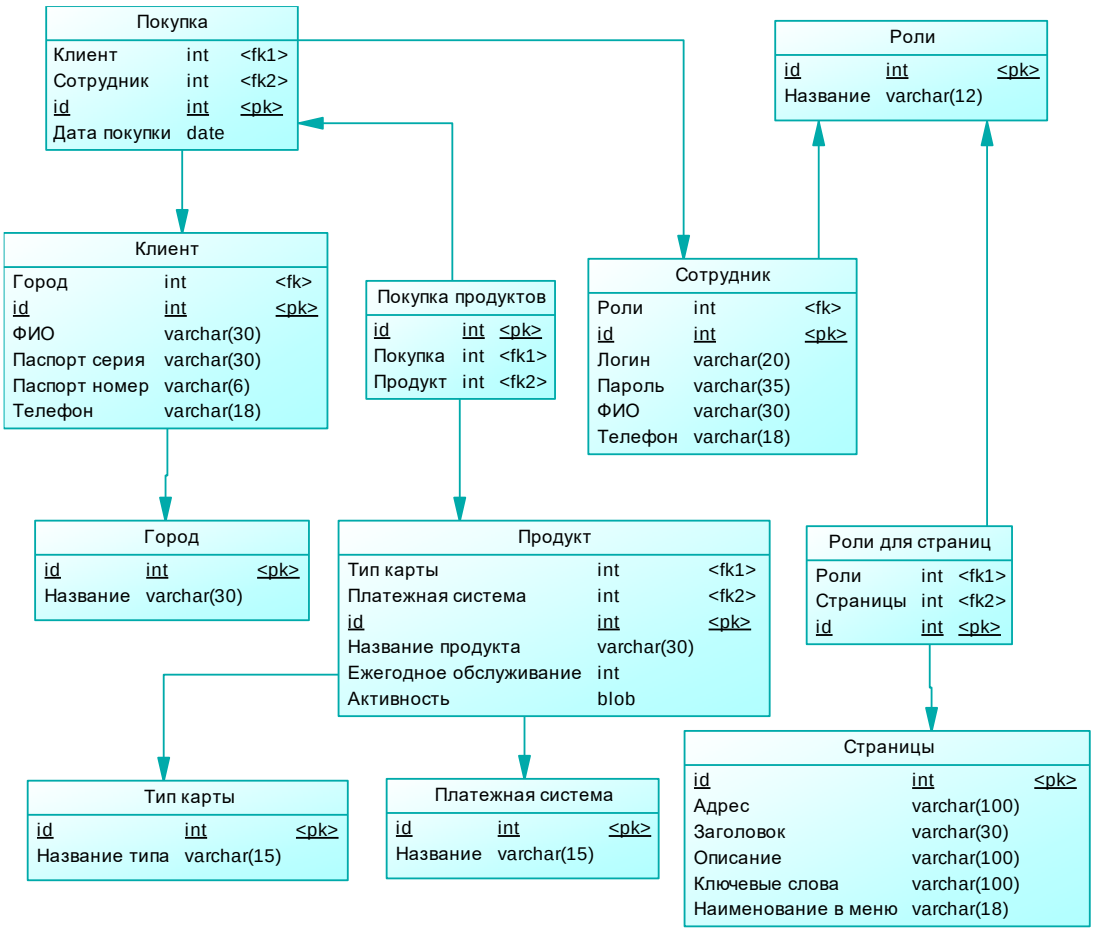

Рисунок 4.24 – Физическая модель базы данных

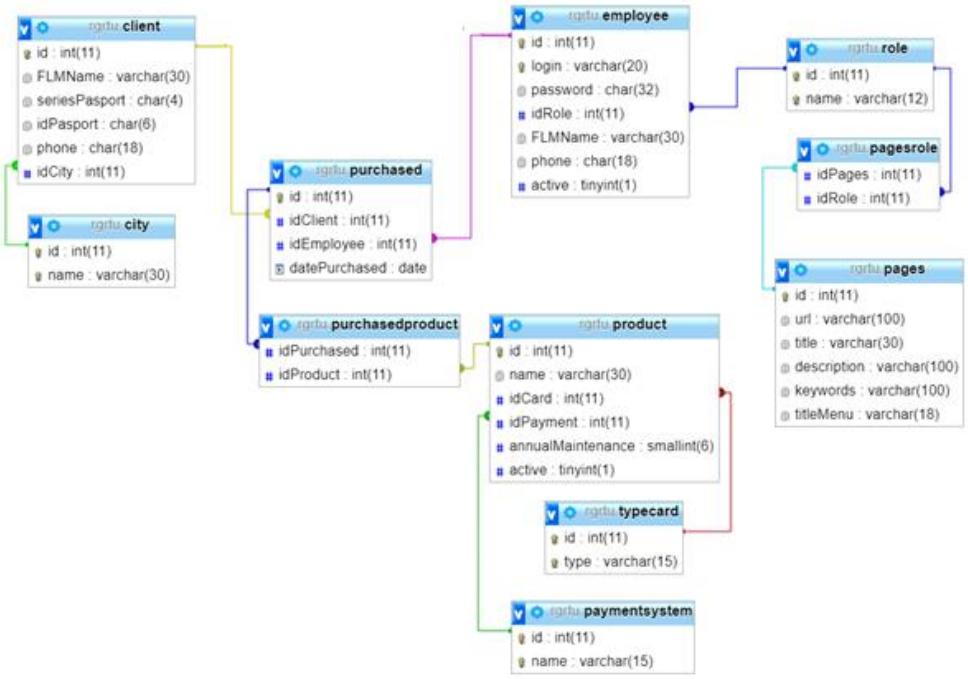

Рисунок 4.25 – Схема развернутой базы данных

4.4 Разработка пользовательского интерфейса

Создание дизайна интерфейса очень важный этап разработки информационной системы. На этом этапе создавались шаблоны страниц сайта в формате PSD-файлов в программе Photoshop. В целях упрощения процесса формирования дизайна сайта, и в виду его однотипности, нужен шаблон следующих страниц:

- Входа пользователя;
- Любой основной страницы;
- Недостаточно прав;
- 404 ошибка.

Соответственно, дизайны нужны для мобильной и десктопной версии.

Полученный результат показан на рисунках ниже. Остальные страницы будут реализованы по аналогии и не нуждаются в отрисовке.

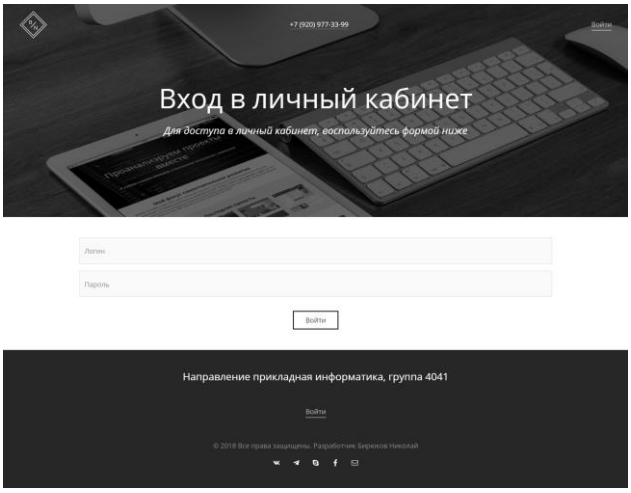

Рисунок 4.26 – Дизайн страницы «Входа пользователя», десктопная версия

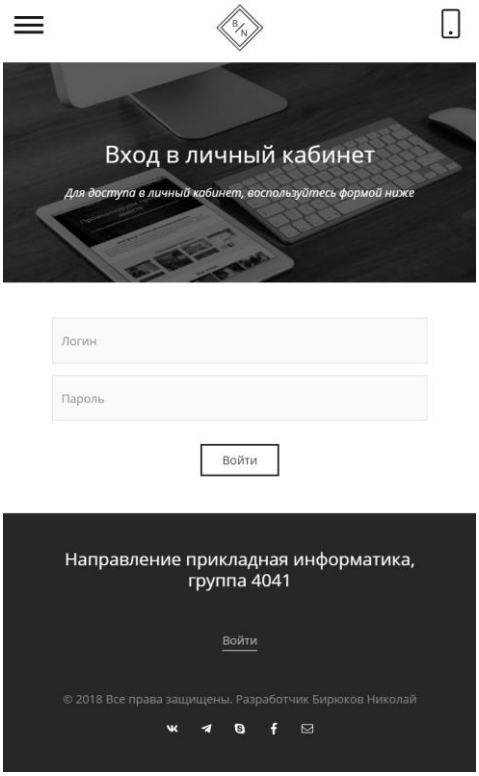

Рисунок 4.27 – Дизайн страницы «Входа пользователя», мобильная версия

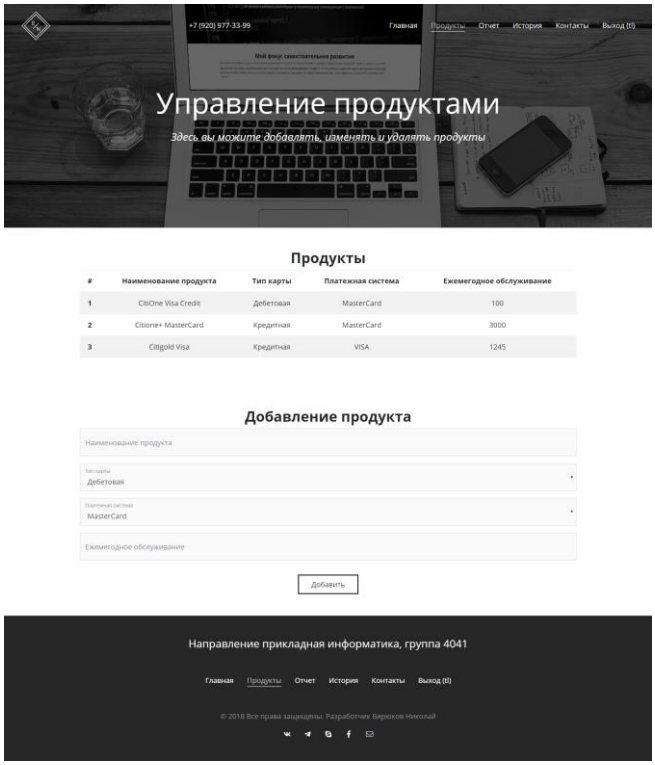

Рисунок 4.28 – Дизайн основных страницы, десктопная версия

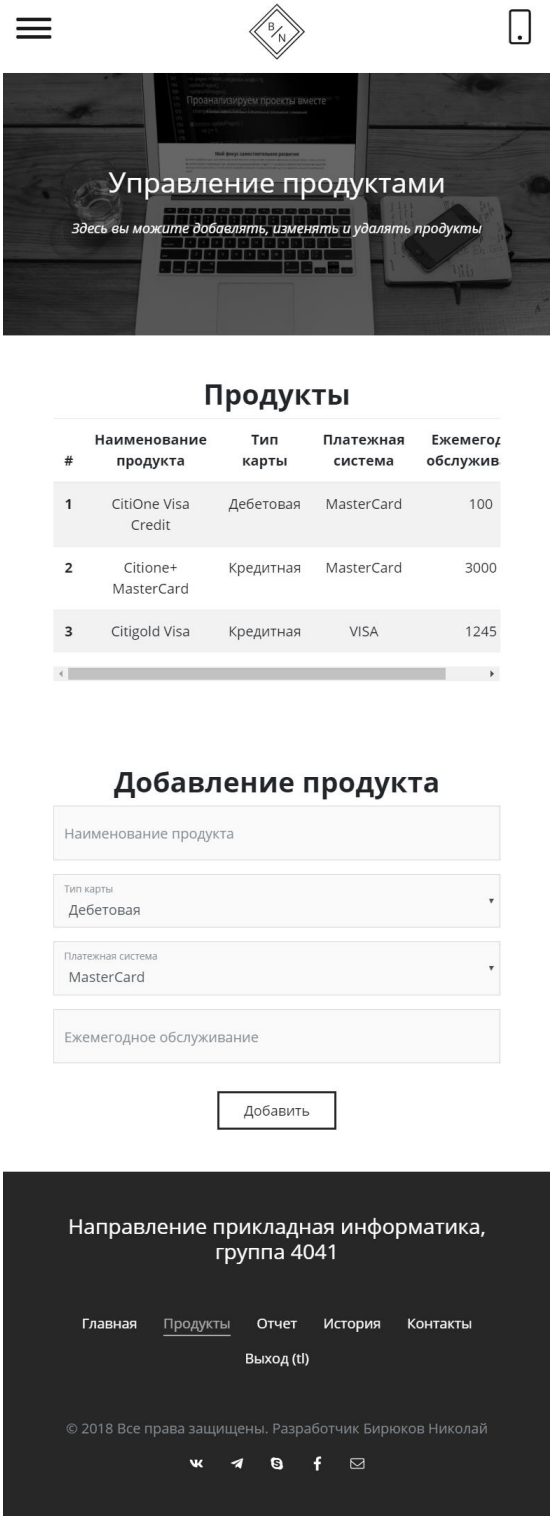

Рисунок 4.29 – Дизайн основных страницы, мобильная версия

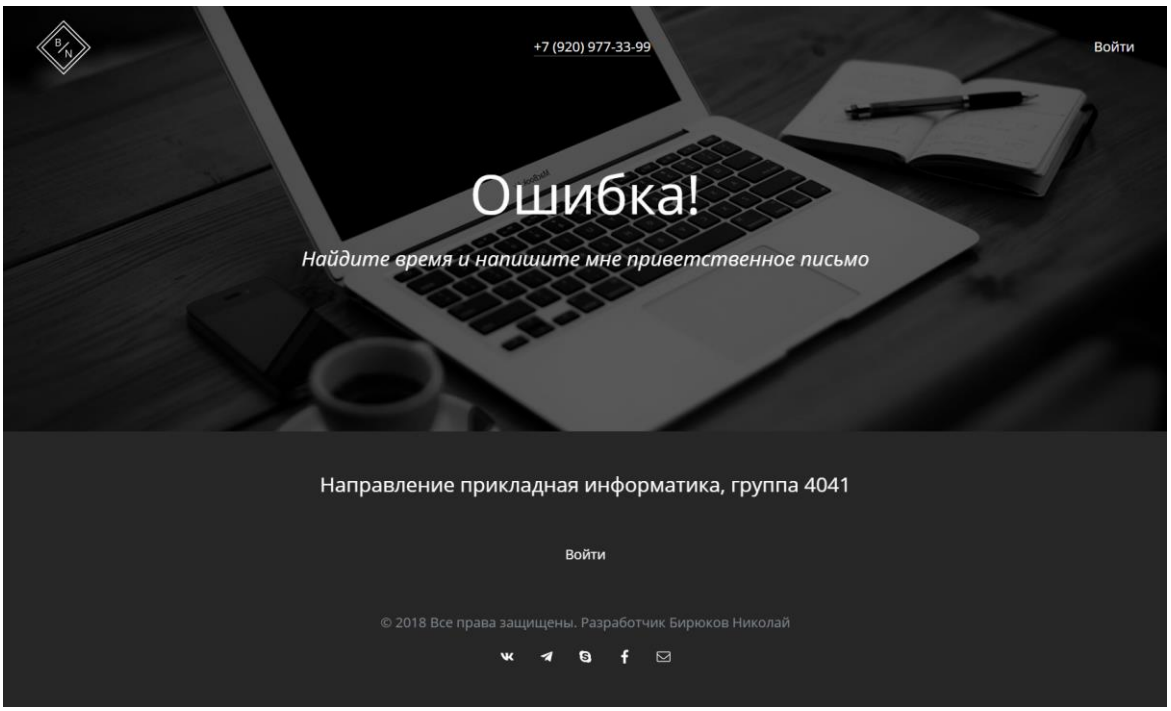

Рисунок 4.30 – Дизайн «404 ошибки», мобильная версия

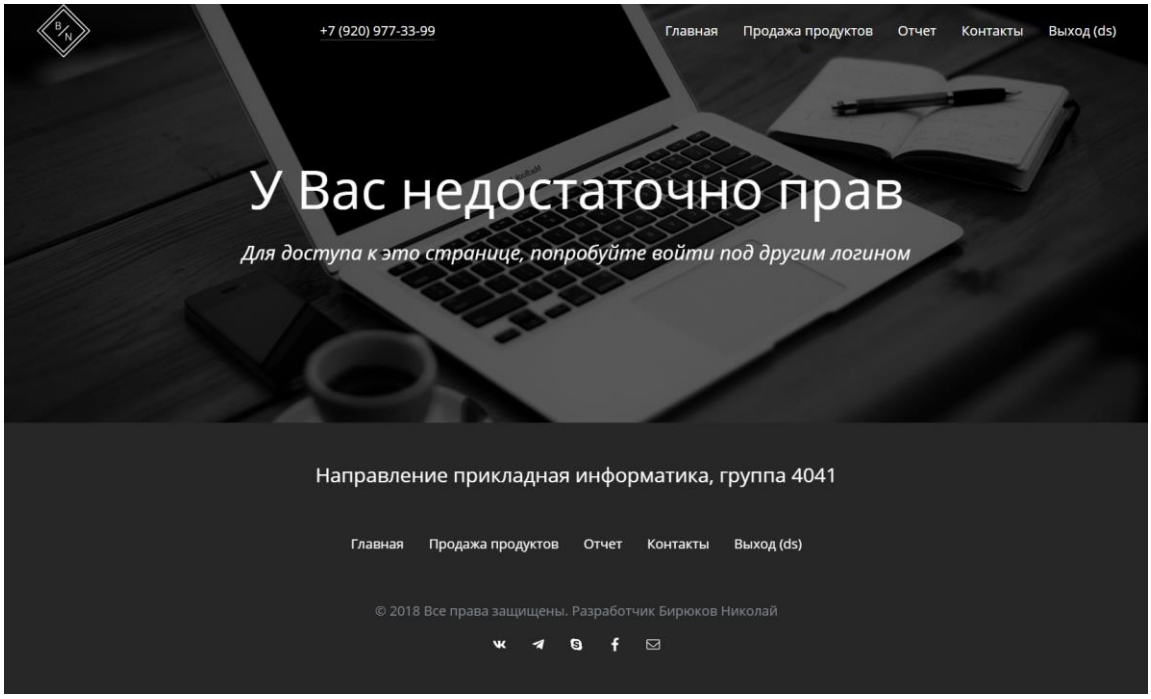

Рисунок 4.31 – Дизайн страницы «Недостаточно прав», мобильная версия

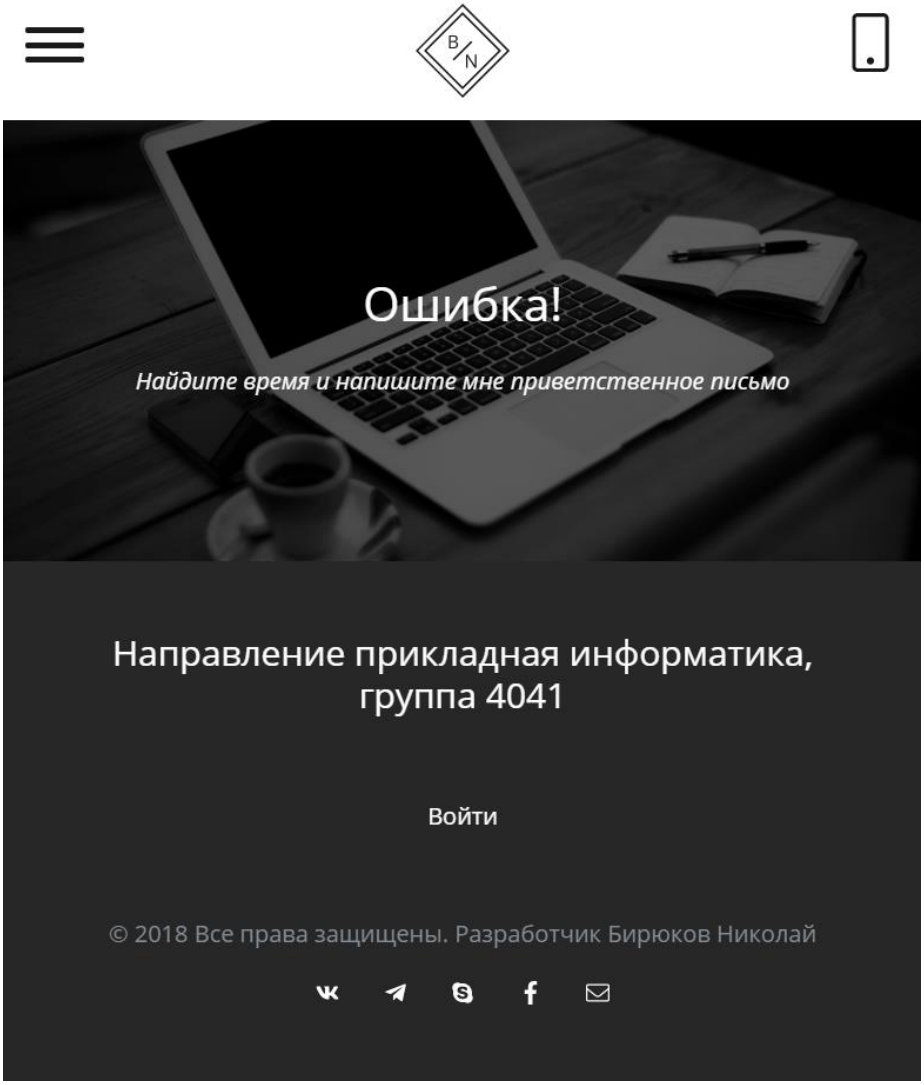

Рисунок 4.32 – Дизайн «404 ошибки»,
мобильная версия

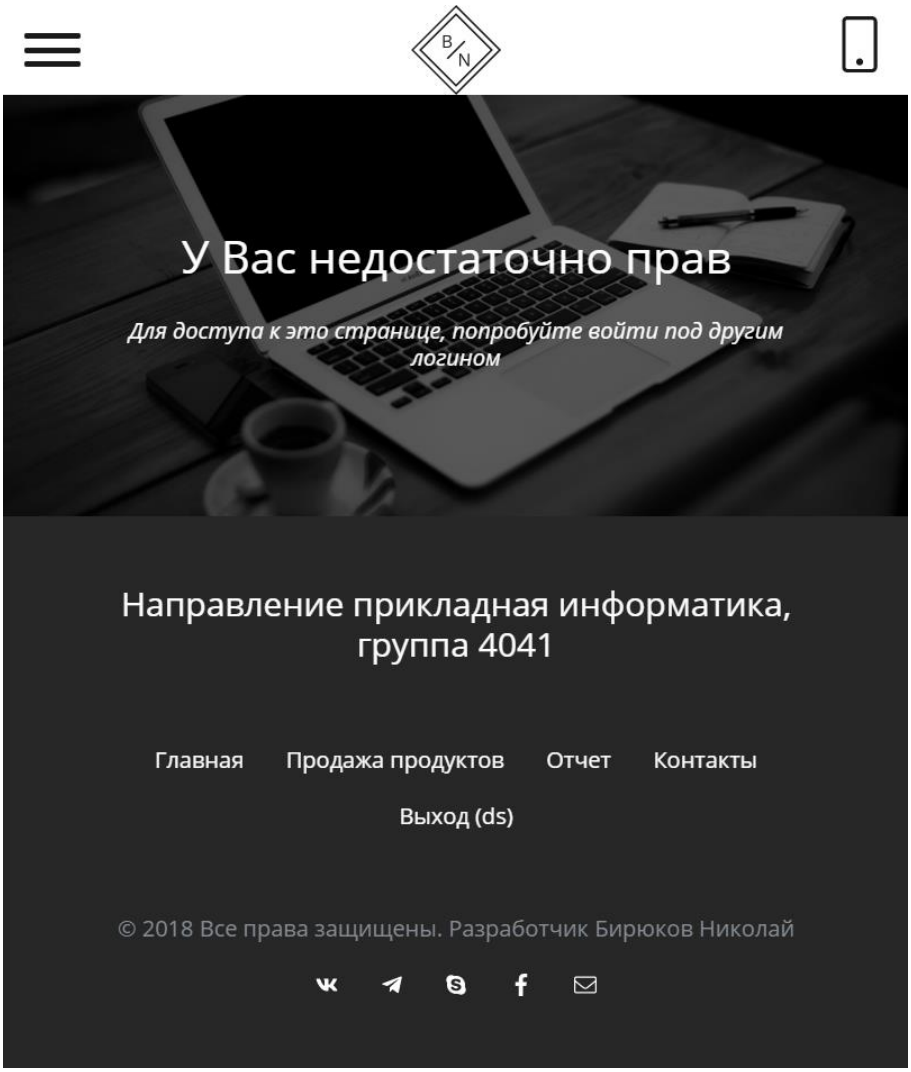

Рисунок 4.33 – Дизайн страницы «Недостаточно прав», мобильная версия

5 ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ

5.1 Назначение системы

Квалификационная работа представляет собой информационную систему, которая состоит из базы данных и программной части.

Основным назначением информационной системы является автоматизации предоставления услуг банком. Для проведения тестирования в базу были добавлены тестовые данные.

5.2 Условия применения

Требования к программным средствам, используемым веб-приложением:

- Операционная система Unix (RHEL, Debian и прочие);
- Веб-сервер Apache 2.4 и выше или Nginx;
- PHP 5.5 и выше (может быть собран как модуль Apache или как CGI-скрипт);
- СУБД MySQL 8 и выше;

Клиентская часть системы требует только установки веб-браузера на компьютере пользователя. Во всех современных операционных системах браузеры входят в стандартную комплектацию системы.

5.3 Руководство оператора

Информационная система может быть запущена с веб-браузера клиентского компьютера. Ее основное назначение состоит в автоматизации предоставления услуг банком.

Информационная система имеет клиент-серверную структуру. Для хранения информации используется база данных.

Для начала работы с системой необходимо войти, для этого необходимо в соответствующие поля ввести выданный «Логин» и «Пароль» и нажать кнопку «Войти».

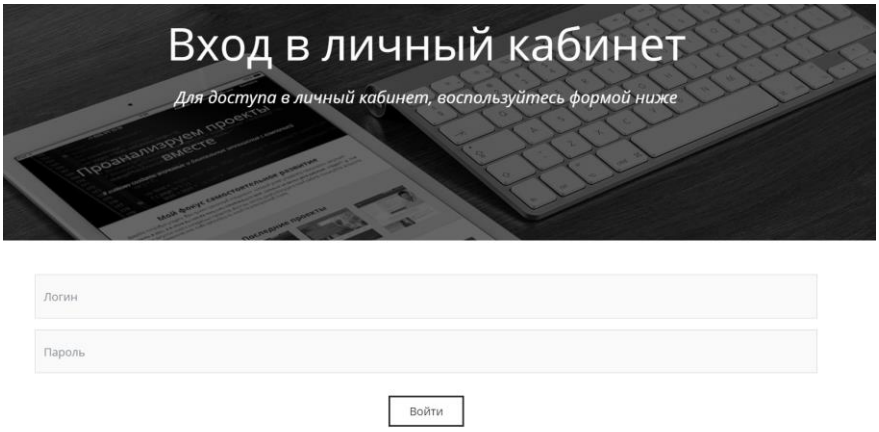

Рисунок 5.1 – Вход в систему

После авторизации откроется интерфейс системы. Основные блоки в интерфейсе показаны на рисунке ниже. **Блок 1** – логотип. **Блок 2** – контактный телефон. **Блок 3** – главное меню системы. **Блок 4** – главный баннер страницы. **Блок 5** – основная часть страницы, это рабочая область где будет происходит работа с сущностями. Блок 6 – нижнее меню системы. **Блок 7** – панель для связи с администратором.

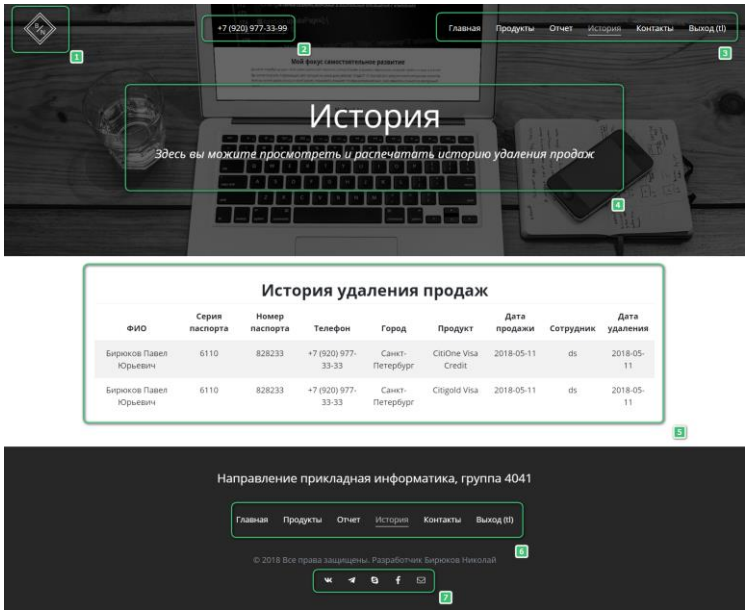

Рисунок 5.2 – Интерфейс системы

Если на странице имеется возможность добавлять информацию, то сразу под **Блоком 1**, будет выведена соответствующая форма. **Блок 2** – обычное поле ввода. **Блок 3** – выпадающий список. **Блок 4** – кнопка действия. После внесения каких-либо изменений, Блок 1 обновится автоматический.

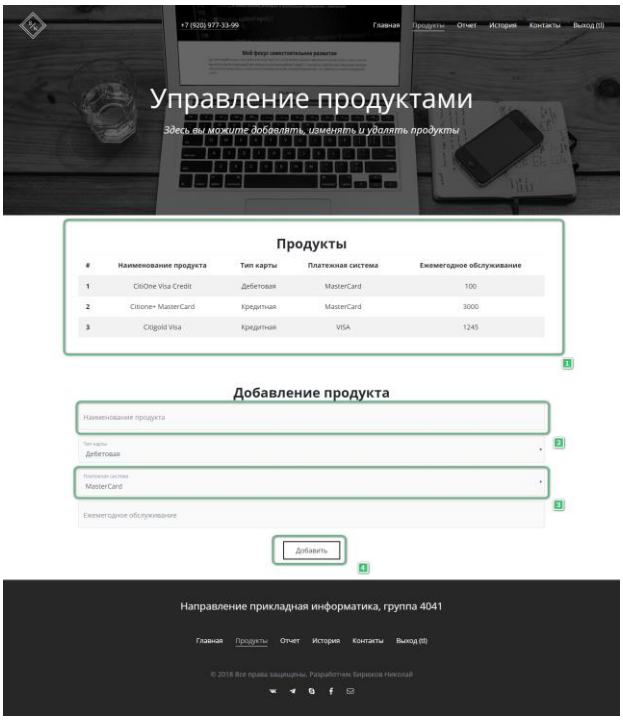

Рисунок 5.3 – Интерфейс системы, форма добавления продукта

6 ТЕСТИРОВАНИЕ СИСТЕМЫ

Разработка web-приложение включает в себя создание большого количества функций, которые в обязательном порядке требуют тестирования и отладки после разработки. Необходимо в первую очередь проверять работоспособность – корректность ссылок, удобство, орфографию и корректное выполнение функций. Важно, чтобы приложение корректно отображался в разных браузерах и различных устройствах.

Заказчик проверяет соответствие структуры сайта техническому заданию, правильность работы функциональных модулей сайта, «полноту» материала, представленного на страницах, правильность отображения в различных браузерах, соответствие кода сайта стандартам кодирования и т.д. При выявлении ошибок производится их немедленное устранение.

Разработанное web-приложение обладает следующими функциями.

Для сотрудников с привилегиями **GIDA**:

- Авторизация по логину и паролю в системе;
- Просмотр главной станицы;
- Управление сотрудниками, добавление, изменение и удаление (делает пользователя неактивным);
- Просмотр страницы с контактами и формой обратной связи.

Для сотрудников с привилегиями **team-leader**:

- Авторизация по логину и паролю в системе;
- Просмотр главной станицы;
- Управление продуктами, добавление, изменение и удаление (делает продукт неактивным, то есть он становится недоступным для продажи);
- Отчет по продажам, доступен выбор периода продажи, а также есть возможность вывода по конкурентному сотруднику, либо сразу по всем;
- История удаленных продаж, доступен выбор периода удаления, а также есть возможность вывода по конкурентному сотруднику, либо сразу по всем;

- Просмотр страницы с контактами и формой обратной связи.

Для сотрудников с привилегиями **DSA**:

- Авторизация по логину и паролю в системе;
- Просмотр главной станицы:
- Управления продажами, оформление, и удаление продажи за текущий день, для удаления доступны продажи пользователя, который зашел в систему;
- Отчет по продажам, доступен выбор периода продажи, отчет выводится по пользователю, который зашел в системы;
- Просмотр страницы с контактами и формой обратной связи.

Также было проведено тестирование целостности базы данных. Проверялась возможность создания новой записи с несуществующим значением внешнего ключа, передача параметров с неверным типов данных или нулевым значением и передача значений, выходящих за допустимую область значений.

Во время прохождения тестирования был выявлен ряд ошибок. Самыми распространенными видами ошибок были:

- Ошибки при подключении к базе данных на начальном этапе разработки ПО.
- Передача не валидных значений в параметры хранимых процедур, отсутствие валидации значений на клиенте.
- Отсутствие обработки возникающих системных исключений.
- Логические ошибки при разработке алгоритма формирования документов.

Большинство ошибок было выявлено и исправлено во время разработки информационной системы. Ошибки пользовательского интерфейса исправлялись также при разработке информационной системы, но уже во время ручного тестирования. Ошибки подключения к базе данных были связаны с неверным указанием параметров подключения.

В результате, все недостатки были устранены и после следующего прохождения тестов все операции были успешно выполнены, ожидаемые результаты полностью совпадали с действительными.

Таким образом, можно сделать вывод, что разработанная система полностью работоспособна и удовлетворяет предъявляемым к ней требованиям.

Заключение

В данной выпускной квалификационной работе была разработана информационная система автоматизации предоставления услуг банком. Был произведен анализ предметной области, на основе которого были сформированы основные требования к системе. С учетом этих требований разработана и протестирована информационная система, способная выполнять поставленные перед ней задачи.

Результатом исследования стала разработанная информационная система, которая использует веб-приложение для продажи продуктов банка.

В ходе выполнения исследования, не удалось подробно познакомится с системами других банков, так как такого рода системы являются внутренними и доступа у обычных пользователей нет. Но было рассмотрено, как происходит оформление продуктов обычными пользователями у банка Тинькофф и Сбербанк.

Инструментами разработки информационной системы явились: Draw.io, HTML, CSS, Bootstrap, JavaScript, jQuery, AJAX, MySQL, PhpMyAdmin.

В разработанной информационной системе используется технология клиент-сервер. Серверная часть включает в себя базу данных. Клиентская часть – графический интерфейс, который размещается на компьютерах пользователей и выполняет описанные выше задачи.

В ходе разработки информационной системы была спроектирована база данных для хранения данных, созданы компоненты авторизации пользователей, внесения новых пользователей, продажи продуктов, вывод истории удаления продаж продуктов и формирование отчетов.

Для организации работы сотрудников банка с информационной системой разработано исчерпывающее руководства пользователя.

Информационная система планируется к внедрению в деятельность банка. С помощью разработанного сервиса можно будет оформить множество продуктов и получить массу положительных отзывов.

Цель данной квалификационной работы можно считать выполненной. В результате была получена работоспособная информационная система, способная выполнять поставленные перед ней задачи.

Список использованных источников

1. Федеральный закон РФ 152-ФЗ «О персональных данных».
2. ГОСТ 19.201-1978. Техническое задание. Требования к содержанию и оформлению.; Введ. 01.01.1990. — М. : Издательство стандартов, 1989. — 13 с.
3. ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы: Гос. стандарт. – Введ. 01.01.1990 // утверждён приказом роспрома 16.09.2004 №95.
4. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Джен нифер Роббинс; [пер. с англ. М. А. Райтман]. — 4-е издание. — М. : Эксмо, 2014. — 528 с.
5. MySQL разработка Web-приложений. – 4-е изд., перераб. И доп. — МПб.: БХВ-Петербург, 2013. — 560 с. : Ил.
6. PHP и MySQL. Исчерпывающее руководство. 2-е изд. / Маклафлин Б. — СПб.: Питер, 2014. — 544 с.: ил.
7. UML: Первое знакомство. / Бабич А.В. – М.: Национальный Открытый Университет «Интуит» — 2016. – 209 с. : ил. ; То жэ [Электронный ресурс]. – URL: http://biblioclub.ru/index.php?page=book_view_red&book_id=233305 (18.04.2018).
8. PHP и JQuery для профиссионалов. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2011. – 352 с. : ил.
9. Бизнес-процессы. – Электронный ресурс. – Режим доступа: http://www.betec.ru (27.05.2018).
10. Панащук С.А. Разработка информационных систем с использованием CASE-систем. "СУБД", 2004, №3.
11. Райордан Р. Основы реляционных баз данных./Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2007. 384с.

12. Савин Р. Тестирование Дот Ком, или Пособие по жесткому обращению с багами в интернет-стартапах. – М.: Дело, 2007. – 312 с.

13. Диего С.М. Проектирование и использования баз данных. – М.: Финансы и статистика, 2003. – 523 с.

14. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). – М., "Лори", 2003.

15. Кузнецов С.Д. Базы данных. Модели и языки. – М.: Бином-Пресс, 2008. – 720 с.

16. Кузнецов С.Д., Проектирование и разработка корпоративных информационных систем. Центр информационных технологий, 2007.

Приложение А

SQL скрипт для создания базы данных

```
-- phpMyAdmin SQL Dump
-- version 4.7.7
-- https://www.phpmyadmin.net/
--
-- Хост: localhost
-- Время создания: Май 14 2018 г., 01:51
-- Версия сервера: 5.7.21-20-beget-5.7.21-20-1-log
-- Версия PHP: 5.6.30

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- База данных: `voronily_rgrtu`
--

-- --------------------------------------------------------

--
-- Структура таблицы `city`
--
-- Создание: Май 13 2018 г., 16:55
-- Последнее обновление: Апр 25 2018 г., 18:23
--

DROP TABLE IF EXISTS `city`;
CREATE TABLE `city` (
  `id` int(11) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `city`
--

INSERT INTO `city` (`id`, `name`) VALUES
(3, 'Москва'),
(4, 'Санкт-Петербург');

-- --------------------------------------------------------

--
-- Структура таблицы `client`
--
-- Создание: Май 07 2018 г., 11:37
-- Последнее обновление: Май 07 2018 г., 11:51
--

DROP TABLE IF EXISTS `client`;
CREATE TABLE `client` (
```

```
  `id` int(11) NOT NULL,
  `FLMName` varchar(30) NOT NULL,
  `seriesPasport` char(4) NOT NULL,
  `idPasport` char(6) NOT NULL,
  `phone` char(18) NOT NULL,
  `idCity` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `client`
--

INSERT INTO `client` (`id`, `FLMName`, `seriesPasport`, `idPasport`, `phone`,
`idCity`) VALUES
(1, 'Иванов Иван Петрович', '6113', '828820', '+7 (920) 977-15-22', 3),
(2, 'Бирюков Павел Юрьевич', '6110', '828233', '+7 (920) 977-33-33', 4),
(4, '2', '2', '2', '2', 3);

-- --------------------------------------------------------

--
-- Структура таблицы `employee`
--
-- Создание: Май 13 2018 г., 17:21
-- Последнее обновление: Май 13 2018 г., 17:21
--

DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `id` int(11) NOT NULL,
  `login` varchar(20) NOT NULL,
  `password` char(32) NOT NULL,
  `idRole` int(11) NOT NULL,
  `FLMName` varchar(30) NOT NULL,
  `phone` char(18) NOT NULL,
  `active` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `employee`
--

INSERT INTO `employee` (`id`, `login`, `password`, `idRole`, `FLMName`,
`phone`, `active`) VALUES
(1, 'gi', 'da0b566359c3862f20b5072c3d49bd0f', 1, 'Бирюков Николай Юрьевич',
'+7 (920) 977-33-99', 1),
(2, 'tl', 'da0b566359c3862f20b5072c3d49bd0f', 2, 'Бирюков Николай Юрьевич',
'+7 (920) 977-33-99', 1),
(3, 'ds', 'da0b566359c3862f20b5072c3d49bd0f', 3, 'Бирюков Николай Юрьевич',
'+7 (920) 977-33-99', 1);

-- --------------------------------------------------------

--
-- Структура таблицы `historyPurchased`
--
-- Создание: Май 13 2018 г., 19:37
-- Последнее обновление: Май 13 2018 г., 21:32
--

DROP TABLE IF EXISTS `historyPurchased`;
CREATE TABLE `historyPurchased` (
  `id` int(11) NOT NULL,
```

```
  `idClient` int(11) NOT NULL,
  `idEmployee` int(11) NOT NULL,
  `datePurchased` date NOT NULL,
  `dateRemoved` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `historyPurchased`
--

INSERT INTO `historyPurchased` (`id`, `idClient`, `idEmployee`,
`datePurchased`, `dateRemoved`) VALUES
(3, 1, 3, '2018-05-11', '2018-05-11'),
(4, 2, 3, '2018-05-11', '2018-05-11'),
(0, 1, 3, '2018-05-11', '2018-05-11'),
(9, 2, 3, '2018-05-11', '2018-05-11'),
(10, 4, 2, '2018-05-11', '2018-05-11');

-- --------------------------------------------------------

--
-- Структура таблицы `historyPurchasedProduct`
--
-- Создание: Май 11 2018 г., 12:56
-- Последнее обновление: Май 11 2018 г., 12:58
--

DROP TABLE IF EXISTS `historyPurchasedProduct`;
CREATE TABLE `historyPurchasedProduct` (
  `idPurchased` int(11) NOT NULL,
  `idProduct` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `historyPurchasedProduct`
--

INSERT INTO `historyPurchasedProduct` (`idPurchased`, `idProduct`) VALUES
(6, 1),
(7, 1),
(7, 2),
(8, 2),
(8, 3),
(9, 1),
(9, 3);

-- --------------------------------------------------------

--
-- Структура таблицы `pages`
--
-- Создание: Май 04 2018 г., 18:29
-- Последнее обновление: Май 13 2018 г., 20:13
--

DROP TABLE IF EXISTS `pages`;
CREATE TABLE `pages` (
  `id` int(11) NOT NULL,
  `url` varchar(100) NOT NULL,
  `title` varchar(30) NOT NULL,
  `description` varchar(100) NOT NULL,
  `keywords` varchar(100) NOT NULL,
  `titleMenu` varchar(18) NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `pages`
--

INSERT INTO `pages` (`id`, `url`, `title`, `description`, `keywords`,
`titleMenu`) VALUES
(1, '/home/', 'Главная', 'Главная', 'РГРТУ, группа 4041', 'Главная'),
(2, '/contact/', 'Контактная информация', 'Контактная информация',
'Контактная информация', 'Контакты'),
(3, '/employee/', 'Управление сотрудниками', 'Управление сотрудниками',
'Управление сотрудниками, добавление сотрудников, изменение сотрудников,
удаление сотрудников', 'Сотрудники'),
(4, '/product/', 'Управление продуктами', 'Управление продуктами', 'Продукты
банка, банковские продукты', 'Продукты'),
(5, '/report/', 'Отчет по продажам', 'Отчет по продажам', 'Отчет по проданным
продуктам', 'Отчет'),
(6, '/history/', 'История продаж', 'История продаж', 'История продаж',
'История'),
(7, '/purchased/', 'Продажа продуктов', 'Продажа продуктов', 'Продажа
продуктов', 'Продажа продуктов');

-- --------------------------------------------------------

--
-- Структура таблицы `pagesRole`
--
-- Создание: Май 09 2018 г., 15:03
-- Последнее обновление: Май 13 2018 г., 20:09
--

DROP TABLE IF EXISTS `pagesRole`;
CREATE TABLE `pagesRole` (
  `idPages` int(11) NOT NULL,
  `idRole` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `pagesRole`
--

INSERT INTO `pagesRole` (`idPages`, `idRole`) VALUES
(1, 1),
(3, 1),
(2, 1),
(1, 2),
(4, 2),
(5, 2),
(6, 2),
(2, 2),
(1, 3),
(7, 3),
(5, 3),
(2, 3);

-- --------------------------------------------------------

--
-- Структура таблицы `paymentSystem`
--
-- Создание: Май 13 2018 г., 16:56
-- Последнее обновление: Апр 25 2018 г., 18:55
```

```
--

DROP TABLE IF EXISTS `paymentSystem`;
CREATE TABLE `paymentSystem` (
  `id` int(11) NOT NULL,
  `name` varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `paymentSystem`
--

INSERT INTO `paymentSystem` (`id`, `name`) VALUES
(1, 'MasterCard'),
(2, 'VISA'),
(3, 'МИР');

-- --------------------------------------------------------

--
-- Структура таблицы `product`
--
-- Создание: Май 13 2018 г., 19:26
-- Последнее обновление: Май 13 2018 г., 19:28
--

DROP TABLE IF EXISTS `product`;
CREATE TABLE `product` (
  `id` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `idCard` int(11) NOT NULL,
  `idPayment` int(11) NOT NULL,
  `annualMaintenance` smallint(6) NOT NULL,
  `active` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `product`
--

INSERT INTO `product` (`id`, `name`, `idCard`, `idPayment`,
`annualMaintenance`, `active`) VALUES
(1, 'CitiOne Visa Credit', 1, 1, 100, 1),
(2, 'Citione+ MasterCard', 2, 1, 3000, 1),
(3, 'Citigold Visa', 2, 2, 1245, 1);

-- --------------------------------------------------------

--
-- Структура таблицы `purchased`
--
-- Создание: Май 10 2018 г., 12:18
-- Последнее обновление: Май 13 2018 г., 21:31
--

DROP TABLE IF EXISTS `purchased`;
CREATE TABLE `purchased` (
  `id` int(11) NOT NULL,
  `idClient` int(11) NOT NULL,
  `idEmployee` int(11) NOT NULL,
  `datePurchased` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
--
-- Дамп данных таблицы `purchased`
--

INSERT INTO `purchased` (`id`, `idClient`, `idEmployee`, `datePurchased`)
VALUES
(1, 2, 3, '2018-04-25'),
(2, 2, 3, '2018-04-25'),
(3, 2, 3, '2018-04-25');

--
-- Триггеры `purchased`
--
DROP TRIGGER IF EXISTS `historyRemovedPurchased`;
DELIMITER $$
CREATE TRIGGER `historyRemovedPurchased` BEFORE DELETE ON `purchased` FOR
EACH ROW BEGIN
    INSERT INTO `historyPurchased`(
        `id`,
        `idClient`,
        `idEmployee`,
        `datePurchased`,
        `dateRemoved`
    )
VALUES(
    OLD.id,
    OLD.idClient,
    OLD.idEmployee,
    OLD.datePurchased,
    CURDATE()
);
INSERT INTO `historyPurchasedProduct`(`idPurchased`, `idProduct`)
SELECT
    `idPurchased`,
    `idProduct`
FROM
    `purchasedProduct`
WHERE
    `idPurchased` = OLD.id;
END
$$
DELIMITER ;

-- --------------------------------------------------------

--
-- Структура таблицы `purchasedProduct`
--
-- Создание: Май 13 2018 г., 17:22
-- Последнее обновление: Май 13 2018 г., 17:22
--

DROP TABLE IF EXISTS `purchasedProduct`;
CREATE TABLE `purchasedProduct` (
  `idPurchased` int(11) NOT NULL,
  `idProduct` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `purchasedProduct`
--

INSERT INTO `purchasedProduct` (`idPurchased`, `idProduct`) VALUES
```

```
(2, 2),
(2, 1),
(3, 3),
(1, 2);

-- --------------------------------------------------------

--
-- Структура таблицы `role`
--
-- Создание: Май 13 2018 г., 16:56
-- Последнее обновление: Май 04 2018 г., 19:31
--

DROP TABLE IF EXISTS `role`;
CREATE TABLE `role` (
  `id` int(11) NOT NULL,
  `name` varchar(12) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `role`
--

INSERT INTO `role` (`id`, `name`) VALUES
(3, 'dsa'),
(1, 'gida'),
(2, 'team-leader');

-- --------------------------------------------------------

--
-- Структура таблицы `typeCard`
--
-- Создание: Май 13 2018 г., 16:56
-- Последнее обновление: Май 07 2018 г., 12:10
--

DROP TABLE IF EXISTS `typeCard`;
CREATE TABLE `typeCard` (
  `id` int(11) NOT NULL,
  `type` varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `typeCard`
--

INSERT INTO `typeCard` (`id`, `type`) VALUES
(1, 'Дебетовая'),
(2, 'Кредитная');

--
-- Индексы сохранённых таблиц
--

--
-- Индексы таблицы `city`
--
ALTER TABLE `city`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `name` (`name`);
```

```
--
-- Индексы таблицы `client`
--
ALTER TABLE `client`
  ADD PRIMARY KEY (`id`),
  ADD KEY `idCity` (`idCity`) USING BTREE;

--
-- Индексы таблицы `employee`
--
ALTER TABLE `employee`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `login` (`login`),
  ADD KEY `idRole` (`idRole`) USING BTREE;

--
-- Индексы таблицы `historyPurchased`
--
ALTER TABLE `historyPurchased`
  ADD KEY `idClient` (`idClient`),
  ADD KEY `idEmployee` (`idEmployee`);

--
-- Индексы таблицы `pages`
--
ALTER TABLE `pages`
  ADD PRIMARY KEY (`id`);

--
-- Индексы таблицы `pagesRole`
--
ALTER TABLE `pagesRole`
  ADD KEY `idPages` (`idPages`) USING BTREE,
  ADD KEY `idRole` (`idRole`) USING BTREE;

--
-- Индексы таблицы `paymentSystem`
--
ALTER TABLE `paymentSystem`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `name` (`name`);

--
-- Индексы таблицы `product`
--
ALTER TABLE `product`
  ADD PRIMARY KEY (`id`),
  ADD KEY `idCard` (`idCard`),
  ADD KEY `idPayment` (`idPayment`) USING BTREE;

--
-- Индексы таблицы `purchased`
--
ALTER TABLE `purchased`
  ADD PRIMARY KEY (`id`),
  ADD KEY `idClient` (`idClient`) USING BTREE,
  ADD KEY `idEmployee` (`idEmployee`) USING BTREE;

--
-- Индексы таблицы `purchasedProduct`
--
ALTER TABLE `purchasedProduct`
  ADD KEY `IDPurchased` (`idPurchased`),
```

```
  ADD KEY `IDProduct` (`idProduct`);

--
-- Индексы таблицы `role`
--
ALTER TABLE `role`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `name` (`name`);

--
-- Индексы таблицы `typeCard`
--
ALTER TABLE `typeCard`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `type` (`type`);

--
-- AUTO_INCREMENT для сохранённых таблиц
--

--
-- AUTO_INCREMENT для таблицы `city`
--
ALTER TABLE `city`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT для таблицы `client`
--
ALTER TABLE `client`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT для таблицы `employee`
--
ALTER TABLE `employee`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=14;

--
-- AUTO_INCREMENT для таблицы `pages`
--
ALTER TABLE `pages`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

--
-- AUTO_INCREMENT для таблицы `paymentSystem`
--
ALTER TABLE `paymentSystem`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO_INCREMENT для таблицы `product`
--
ALTER TABLE `product`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

--
-- AUTO_INCREMENT для таблицы `purchased`
--
ALTER TABLE `purchased`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
```

```
-- AUTO_INCREMENT для таблицы `role`
--
ALTER TABLE `role`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO INCREMENT для таблицы `typeCard`
--
ALTER TABLE `typeCard`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- Ограничения внешнего ключа сохраненных таблиц
--

--
-- Ограничения внешнего ключа таблицы `client`
--
ALTER TABLE `client`
  ADD CONSTRAINT `client_ibfk_1` FOREIGN KEY (`idCity`) REFERENCES `city`
(`id`);

--
-- Ограничения внешнего ключа таблицы `employee`
--
ALTER TABLE `employee`
  ADD CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`idRole`) REFERENCES `role`
(`id`);

--
-- Ограничения внешнего ключа таблицы `historyPurchased`
--
ALTER TABLE `historyPurchased`
  ADD CONSTRAINT `historyPurchased_ibfk_1` FOREIGN KEY (`idClient`)
REFERENCES `client` (`id`),
  ADD CONSTRAINT `historyPurchased_ibfk_2` FOREIGN KEY (`idEmployee`)
REFERENCES `employee` (`id`);

--
-- Ограничения внешнего ключа таблицы `pagesRole`
--
ALTER TABLE `pagesRole`
  ADD CONSTRAINT `pagesRole_ibfk_1` FOREIGN KEY (`idPages`) REFERENCES
`pages` (`id`) ON UPDATE CASCADE,
  ADD CONSTRAINT `pagesRole_ibfk_2` FOREIGN KEY (`idRole`) REFERENCES `role`
(`id`);

--
-- Ограничения внешнего ключа таблицы `product`
--
ALTER TABLE `product`
  ADD CONSTRAINT `product_ibfk_1` FOREIGN KEY (`idCard`) REFERENCES
`typeCard` (`id`),
  ADD CONSTRAINT `product_ibfk_2` FOREIGN KEY (`idPayment`) REFERENCES
`paymentSystem` (`id`);

--
-- Ограничения внешнего ключа таблицы `purchased`
--
ALTER TABLE `purchased`
  ADD CONSTRAINT `purchased_ibfk_1` FOREIGN KEY (`idClient`) REFERENCES
`client` (`id`),
  ADD CONSTRAINT `purchased_ibfk_2` FOREIGN KEY (`idEmployee`) REFERENCES
```

```
`employee` (`id`);

--
-- Ограничения внешнего ключа таблицы `purchasedProduct`
--
ALTER TABLE `purchasedProduct`
  ADD CONSTRAINT `purchasedProduct_ibfk_1` FOREIGN KEY (`idPurchased`)
REFERENCES `purchased` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `purchasedProduct_ibfk_2` FOREIGN KEY (`idProduct`)
REFERENCES `product` (`id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Приложение Б

Основной скрипт API

```
<?php
define("_ACCESS_INCLUDE", true);
include_once _ROOT . "/include/sql.php";
session_start();
$action = $_GET["action"];

class purchasedProduct
{
    public function __construct()
    {
    }

    public function signIn()
    {
        $login = $_POST["login"];
        $password = md5($_POST["password"]);
        $pageRedirect = (isset($_GET["page"])) ? urldecode($_GET["page"]) :
"/";
        $sql = "SELECT
                    `employee`.`id` AS `id`,
                    `role`.`name` AS `role`
                FROM
                    `employee`
                INNER JOIN `role` ON `role`.`id` = `employee`.`idRole`
                WHERE
                    `login` = '$login' AND `password` = '$password'";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);

        if (!$row = $result->fetch_assoc()) {
            $resultArr = [
                "status" => false,
                "description" => "<strong>Ошибка!</strong> Проверьте
правильность ввода данных"
            ];
            return json_encode($resultArr);
        };

        $_SESSION["id"] = $row["id"];
        $_SESSION["login"] = $login;
        $_SESSION["role"] = $row["role"];

        $resultArr = [
            "status" => true,
            "action" => "sign-in",
            "redirect" => $pageRedirect
        ];
        return json_encode($resultArr);
    }

    public function getEmployee()
    {
        $sql = "SELECT
                    `employee`.`id` AS `id`,
                    `employee`.`login` AS `login`,
                    `employee`.`FLMname` AS `FLMname`,
                    `employee`.`phone` AS `phone`,
                    `role`.`name` AS `role`
```

```
                FROM
                    `employee`
                INNER JOIN `role` ON `role`.`id` = `employee`.`idRole`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);

        $html = "<table class=\"table table-striped\">
                    <thead>
                    <tr>
                        <th class=\"text-center\">#</th>
                        <th class=\"text-center\">Логин</th>
                        <th class=\"text-center\">Пароль</th>
                        <th class=\"text-center\">Роль</th>
                        <th class=\"text-center\">ФИО</th>
                        <th class=\"text-center\">Телефон</th>
                    </tr>
                    </thead>
                    <tbody>";

        while ($row = $result->fetch_assoc()) {
            $html .= "<tr>
                            <th class=\"text-center\">{$row["id"]}</th>
                            <td class=\"text-center\">{$row["login"]}</td>
                            <td class=\"text-center\">**********</td>
                            <td class=\"text-center\">{$row["role"]}</td>
                            <td class=\"text-center\">{$row["FLMname"]}</td>
                            <td class=\"text-center\">{$row["phone"]}</td>
                      </tr>";
        }
        $html .= "</tbody>
                  </table>";

        $resultArr = [
            "status" => true,
            "action" => "get-employee",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function getProduct()
    {
        $sql = "SELECT
                    `product`.`id` AS `id`,
                    `product`.`name` AS `name`,
                    `product`.`annualMaintenance` AS `annualMaintenance`,
                    `typeCard`.`type` AS `type`,
                    `paymentSystem`.`name` AS `system`
                FROM
                    `product`
                INNER JOIN `typeCard` ON `typeCard`.`id` = `product`.`idCard`
                INNER JOIN `paymentSystem` ON `paymentSystem`.`id` =
`product`.`idPayment`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);

        $html = "<table class=\"table table-striped\">
                    <thead>
                    <tr>
                        <th class=\"text-center\">#</th>
                        <th class=\"text-center\">Наименование продукта</th>
                        <th class=\"text-center\">Тип карты</th>
                        <th class=\"text-center\">Платежная система</th>
```

```
                        <th class=\"text-center\">Ежемегодное
обслуживание</th>
                    </tr>
                    </thead>
                    <tbody>";

        while ($row = $result->fetch_assoc()) {
            $html .= "<tr>
                            <th class=\"text-center\">{$row["id"]}</th>
                            <td class=\"text-center\">{$row["name"]}</td>
                            <td class=\"text-center\">{$row["type"]}</td>
                            <td class=\"text-center\">{$row["system"]}</td>
                            <td class=\"text-
center\">{$row["annualMaintenance"]}</td>
                        </tr>";
        }
        $html .= "</tbody>
                  </table>";

        $resultArr = [
            "status" => true,
            "action" => "get-product",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function getReport()
    {
        $id = $_POST["id"];
        $dateStart = (!isset($_POST["dateStart"])) ? "1990-01-01" :
$_POST["dateStart"];
        $dateEnd = (!isset($_POST["dateEnd"])) ? "2020-01-01" :
$_POST["dateEnd"];
        $whereId = (!isset($id) || $id === "all") ? "" : "`employee`.`id` =
'{$id}' AND ";
        $sql = "SELECT
                    `purchased`.`id` AS `id`,
                    `purchased`.`datePurchased` AS `date`,
                    `client`.`FLMName` AS `FLMName`,
                    `client`.`seriesPasport` AS `seriesPasport`,
                    `client`.`idPasport` AS `idPasport`,
                    `client`.`phone` AS `phone`,
                    `employee`.`login` AS `login`,
                    `city`.`name` AS `city`,
                    GROUP_CONCAT(`product`.`name` SEPARATOR '; <br />') AS
`products`
                FROM
                    `purchased`
                INNER JOIN `client` ON `client`.`id` = `purchased`.`idClient`
                INNER JOIN `employee` ON `employee`.`id` =
`purchased`.`idEmployee`
                INNER JOIN `city` ON `city`.`id` = `client`.`idCity`
                INNER JOIN `purchasedProduct` ON
`purchasedProduct`.`idPurchased` = `purchased`.`id`
                INNER JOIN `product` ON `product`.`id` =
`purchasedProduct`.`idProduct`
                WHERE
                    {$whereId}`purchased`.`datePurchased` >= '{$dateStart}'
AND `purchased`.`datePurchased` <= '{$dateEnd}'
                GROUP BY
                    `purchased`.`id`";
```

```
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);

        $html = "<table class=\"table table-striped\">
                    <thead>
                    <tr>
                        <th class=\"text-center\">#</th>
                        <th class=\"text-center\">ФИО</th>
                        <th class=\"text-center\">Серия паспорта</th>
                        <th class=\"text-center\">Номер паспорта</th>
                        <th class=\"text-center\">Телефон</th>
                        <th class=\"text-center\">Город</th>
                        <th class=\"text-center\">Продукты</th>
                        <th class=\"text-center\">Дата продажи</th>
                        <th class=\"text-center\">Сотрудник</th>
                    </tr>
                    </thead>
                    <tbody>";

        while ($row = $result->fetch_assoc()) {
            $html .= "<tr>
                            <th class=\"text-center\">{$row["id"]}</th>
                            <td class=\"text-center\">{$row["FLMName"]}</td>
                            <td class=\"text-
center\">{$row["seriesPasport"]}</td>
                            <td class=\"text-center\">{$row["idPasport"]}</td>
                            <td class=\"text-center\">{$row["phone"]}</td>
                            <td class=\"text-center\">{$row["city"]}</td>
                            <td class=\"text-center\">{$row["products"]}</td>
                            <td class=\"text-center\">{$row["date"]}</td>
                            <td class=\"text-center\">{$row["login"]}</td>
                      </tr>";
        }
        $html .= "</tbody>
                  </table>";

        $resultArr = [
            "status" => true,
            "action" => "get-report",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function getHistory() {
        $sql = "SELECT
                    `historyPurchased`.`datePurchased` AS `date`,
                    `historyPurchased`.`dateRemoved` AS `dateRemoved`,
                    `client`.`FLMName` AS `FLMName`,
                    `client`.`seriesPasport` AS `seriesPasport`,
                    `client`.`idPasport` AS `idPasport`,
                    `client`.`phone` AS `phone`,
                    `employee`.`login` AS `login`,
                    `city`.`name` AS `city`,
                    `product`.`name` AS `product`
                FROM
                    `historyPurchased`
                INNER JOIN `client` ON `client`.`id` =
`historyPurchased`.`idClient`
                INNER JOIN `employee` ON `employee`.`id` =
`historyPurchased`.`idEmployee`
                INNER JOIN `city` ON `city`.`id` = `client`.`idCity`
                INNER JOIN `historyPurchasedProduct` ON
```

```
`historyPurchasedProduct`.`idPurchased` = `historyPurchased`.`id`
                INNER JOIN `product` ON `product`.`id` =
`historyPurchasedProduct`.`idProduct`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);

        $html = "<table class=\"table table-striped table-responsive\">
                    <thead>
                    <tr>
                        <th class=\"text-center\">ФИО</th>
                        <th class=\"text-center\">Серия паспорта</th>
                        <th class=\"text-center\">Номер паспорта</th>
                        <th class=\"text-center\">Телефон</th>
                        <th class=\"text-center\">Город</th>
                        <th class=\"text-center\">Продукт</th>
                        <th class=\"text-center\">Дата продажи</th>
                        <th class=\"text-center\">Сотрудник</th>
                        <th class=\"text-center\">Дата удаления</th>
                    </tr>
                    </thead>
                    <tbody>";

        while ($row = $result->fetch_assoc()) {
            $html .= "<tr>
                            <td class=\"text-center\">{$row["FLMName"]}</td>
                            <td class=\"text-
center\">{$row["seriesPasport"]}</td>
                            <td class=\"text-center\">{$row["idPasport"]}</td>
                            <td class=\"text-center\">{$row["phone"]}</td>
                            <td class=\"text-center\">{$row["city"]}</td>
                            <td class=\"text-center\">{$row["product"]}</td>
                            <td class=\"text-center\">{$row["date"]}</td>
                            <td class=\"text-center\">{$row["login"]}</td>
                            <td class=\"text-
center\">{$row["dateRemoved"]}</td>
                        </tr>";
        }
        $html .= "</tbody>
                  </table>";

        $resultArr = [
            "status" => true,
            "action" => "get-purchased-product",
            "html" => $html
        ];
        return json_encode($resultArr);

    }

    public function getPurchased()
    {
        $sql = "SELECT
                    `purchased`.`id` AS `id`,
                    `purchased`.`datePurchased` AS `date`,
                    `client`.`FLMName` AS `FLMName`,
                    `client`.`seriesPasport` AS `seriesPasport`,
                    `client`.`idPasport` AS `idPasport`,
                    `client`.`phone` AS `phone`,
                    `employee`.`login` AS `login`,
                    `city`.`name` AS `city`,
                    `product`.`name` AS `product`
                FROM
                    `purchased`
```

```
                INNER JOIN `client` ON `client`.`id` = `purchased`.`idClient`
                INNER JOIN `employee` ON `employee`.`id` =
`purchased`.`idEmployee`
                INNER JOIN `city` ON `city`.`id` = `client`.`idCity`
                INNER JOIN `purchasedProduct` ON
`purchasedProduct`.`idPurchased` = `purchased`.`id`
                INNER JOIN `product` ON `product`.`id` =
`purchasedProduct`.`idProduct`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);

        $html = "<table class=\"table table-striped table-responsive\">
                    <thead>
                    <tr>
                        <th class=\"text-center\">#</th>
                        <th class=\"text-center\">ФИО</th>
                        <th class=\"text-center\">Серия паспорта</th>
                        <th class=\"text-center\">Номер паспорта</th>
                        <th class=\"text-center\">Телефон</th>
                        <th class=\"text-center\">Город</th>
                        <th class=\"text-center\">Продукт</th>
                        <th class=\"text-center\">Дата продажи</th>
                        <th class=\"text-center\">Сотрудник</th>
                    </tr>
                    </thead>
                    <tbody>";

        while ($row = $result->fetch_assoc()) {
            $html .= "<tr>
                            <th class=\"text-center\">{$row["id"]}</th>
                            <td class=\"text-center\">{$row["FLMName"]}</td>
                            <td class=\"text-
center\">{$row["seriesPasport"]}</td>
                            <td class=\"text-center\">{$row["idPasport"]}</td>
                            <td class=\"text-center\">{$row["phone"]}</td>
                            <td class=\"text-center\">{$row["city"]}</td>
                            <td class=\"text-center\">{$row["product"]}</td>
                            <td class=\"text-center\">{$row["date"]}</td>
                            <td class=\"text-center\">{$row["login"]}</td>
                      </tr>";
        }
        $html .= "</tbody>
                  </table>";

        $resultArr = [
            "status" => true,
            "action" => "get-purchased-product",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function addEmployee()
    {
        $login = $_POST["login"];
        $password = md5($_POST["password"]);
        $role = $_POST["idRole"];
        $FLMname = $_POST["FLMname"];
        $phone = $_POST["phone"];
        $sql = "INSERT INTO `employee`(
                    `login`,
                    `password`,
                    `idRole`,
```

```
                    `FLMName`,
                    `phone`
                )
                VALUES('{$login}', '{$password}', '{$role}', '$FLMname',
'$phone');";
        $mysqli = new sql();
        $mysqli->sqlQuery($sql);
        $resultArr = [
            "status" => true,
            "action" => "add-employee",
            "description" => "Запись успешно добавлена"
        ];
        return json_encode($resultArr);
    }

    public function addProduct()
    {
        $name = $_POST["name"];
        $idCard = $_POST["idCard"];
        $idPayment = $_POST["idPayment"];
        $annualMaintenance = $_POST["annualMaintenance"];
        $sql = "INSERT INTO `product`(
                    `name`,
                    `idCard`,
                    `idPayment`,
                    `annualMaintenance`
                )
                VALUES('{$name}', '{$idCard}', '$idPayment',
'$annualMaintenance');";
        $mysqli = new sql();
        $mysqli->sqlQuery($sql);
        $resultArr = [
            "status" => true,
            "action" => "add-product",
            "description" => "Запись успешно добавлена"
        ];
        return json_encode($resultArr);
    }

    public function getRoleSelect()
    {
        $sql = "SELECT
                    `role`.`id` AS `id`,
                    `role`.`name` AS `name`
                FROM
                    `role`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);
        $html = "";
        while ($row = $result->fetch_assoc()) {
            $html .= "<option
value=\"{$row["id"]}\">{$row["name"]}</option>";
        }
        $resultArr = [
            "status" => true,
            "action" => "get-role-select",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function getCardSelect()
    {
```

```
        $sql = "SELECT
                    `typeCard`.`id` AS `id`,
                    `typeCard`.`type` AS `type`
                FROM
                    `typeCard`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);
        $html = "";
        while ($row = $result->fetch_assoc()) {
            $html .= "<option
value=\"{$row["id"]}\">{$row["type"]}</option>";
        }
        $resultArr = [
            "status" => true,
            "action" => "get-card-select",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function getPaymentSelect()
    {
        $sql = "SELECT
                    `paymentSystem`.`id` AS `id`,
                    `paymentSystem`.`name` AS `name`
                FROM
                    `paymentSystem`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);
        $html = "";
        while ($row = $result->fetch_assoc()) {
            $html .= "<option
value=\"{$row["id"]}\">{$row["name"]}</option>";
        }
        $resultArr = [
            "status" => true,
            "action" => "get-payment-select",
            "html" => $html
        ];
        return json_encode($resultArr);
    }

    public function getEmployeeSelect()
    {
        $sql = "SELECT
                    `employee`.`id` AS `id`,
                    `employee`.`login` AS `login`
                FROM
                    `employee`";
        $mysqli = new sql();
        $result = $mysqli->sqlQuery($sql);
        $html = "";
        while ($row = $result->fetch_assoc()) {
            $html .= "<option
value=\"{$row["id"]}\">{$row["login"]}</option>";
        }
        $resultArr = [
            "status" => true,
            "action" => "get-employee-select",
            "html" => $html
        ];
        return json_encode($resultArr);
    }
```

```
}

switch ($action) {
    case "sign-in":
        $signIn = new purchasedProduct();
        echo $signIn->signIn();
        break;
    case "get-employee":
        $getEmployee = new purchasedProduct();
        echo $getEmployee->getEmployee();
        break;
    case "add-employee":
        $addEmployee = new purchasedProduct();
        echo $addEmployee->addEmployee();
        break;
    case "get-product":
        $getProduct = new purchasedProduct();
        echo $getProduct->getProduct();
        break;
    case "add-product":
        $addProduct = new purchasedProduct();
        echo $addProduct->addProduct();
        break;
    case "get-report":
        $getReport = new purchasedProduct();
        echo $getReport->getReport();
        break;
    case "get-purchased-product":
        $getPurchasedProduct = new purchasedProduct();
        echo $getPurchasedProduct->getPurchased();
        break;
}

?>
```

Основной JS

```
// start mmenu
(function ($) {
    var $window = $(window);
    var $navBarTop = $('.nav-bar_top');
    var $navBarMenuTop = $('#nav-bar__menu_top');
    var $nabBarMenuTopPlugin = $navBarMenuTop.mmenu({
        'navbar': {
            title: 'Основное меню'
        },
        'navbars': [{
            'position': 'bottom',
            'content': [
                '<a class="h4" target="_blank"
href="https://vk.com/biryukovkolya/">' +
                '<svg class="svg-icon" xmlns="http://www.w3.org/2000/svg">' +
                '<use xlink:href="#vk-icon"/>' +
                '</svg>' +
                '</a>',
                '<a class="h4" target="_blank"
href="https://telegram.me/biryukovkolya">' +
                '<svg class="svg-icon" xmlns="http://www.w3.org/2000/svg">' +
                '<use xlink:href="#telegram-plane-icon"/>' +
                '</svg>' +
                '</a>',
```

```
                    '<a class="h4" href="mailto:biryukovkolya@mail.ru">' +
                    '<svg class="svg-icon" xmlns="http://www.w3.org/2000/svg">' +
                    '<use xlink:href="#envelope-icon"/>' +
                    '</svg>' +
                    '</a>'
                ]
            }],
            'extensions': [
                'pagedim-black'
            ]
        }, {
            clone: true
        });
        var $navBarTopToggle = $('#nav-bar__link_toggle');
        var navBarTopAPI = $nabBarMenuTopPlugin.data('mmenu');

        $window.on('resize', function () {
            if ($(this).outerWidth() < 992) {
                $navBarTop.filter('.nav-bar_dark').removeClass('nav-bar_dark')
            } else {
                $navBarTop.not('.nav-bar_dark').addClass('nav-bar_dark')
            }
        });

        $window.trigger('resize');


        $navBarTopToggle.on('click', function () {
            navBarTopAPI.open();
        });

        navBarTopAPI.bind('open:finish', function () {
            $navBarTopToggle.addClass('open');
        });
        navBarTopAPI.bind('close:finish', function () {
            $navBarTopToggle.removeClass('open');
        });
    })(jQuery);
    // end mmenu


    // start scrollspy
    (function ($) {
        var $navBarTop = $('.nav-bar_top');
        var offsetTopOuterHeight = $navBarTop.offset().top +
    $navBarTop.outerHeight();
        var min = offsetTopOuterHeight + 300;
        var $document = $(document);
        var documentOuterHeight = $document.outerHeight();
        var $widnow = $(window);
        var scrollspyAddClassData = $navBarTop.data('scrollspy-add-class-nav-
    bar');
        var scrollspyRemoveClassData = $navBarTop.data('scrollspy-remove-class-
    nav-bar');
        var scrollspyAnimationClassData = $navBarTop.data('scrollspy-animation-
    class-nav-bar');
        var animationReg = /(^|\s)animation-\S+/g;

        $navBarTop.scrollspy({
            min: min,
            max: documentOuterHeight,
            onEnter: function (element) {
                if ($widnow.outerWidth() < 992) {
```

```
                return false;
            }

            $(element)
                .addClass(scrollspyAddClassData + ' ' +
scrollspyAnimationClassData)
                .removeClass(scrollspyRemoveClassData);
        },
        onLeave: function (element) {
            if ($widnow.outerWidth() < 992) {
                return false;
            }

            $(element).on('animationend.nav-bar', function () {
                $(this)
                    .removeClass(scrollspyAddClassData)
                    .addClass(scrollspyRemoveClassData)
                    .off('animationend.nav-bar');
            });
            $(element).addClass(scrollspyAnimationClassData + ' animation-
reverse');
        }
    });

    $('[data-scrollspy-animation-class-nav-bar]').on('animationend', function
() {
        $(this).removeClass(function (index, cls) {
            var removeClass = (cls.match(animationReg) || []).join(' ');
            return removeClass;
        });
    });

    var $scrollspy = $('[data-scrollspy-class]');
    var $window = $(window);
    $scrollspy.each(function () {
        var offsetTop = $(this).offset().top;
        var min = (offsetTop - $window.outerHeight() > 0) ? offsetTop -
$window.outerHeight() : 0;
        var offsetTopOuterHeight = offsetTop + $(this).outerHeight();
        var scrollspyClassData = $(this).data('scrollspy-class');
        $(this).scrollspy({
            min: min,
            max: offsetTopOuterHeight,
            onEnter: function (element) {
                $(element)
                    .removeClass('invisible')
                    .addClass(scrollspyClassData);
            }
            // onLeave: function (element) {
            //     $(element).filter('.' +
scrollspyClassData).removeClass(scrollspyClassData)
            // }
        });
    });

    $window.trigger('scroll');

})(jQuery);
// end scrollspy


// start inputmask
(function ($) {
```

```
    var $email = $('#email-address');

    $email.inputmask({
        alias: 'email',
        showMaskOnHover: false
    });


    $('#first-name').inputmask({
        regex: '[а-яё]*',
        casing: true,
        showMaskOnHover: false
    });

})(jQuery);
// end inputmask

// start fancybox
(function ($) {
    var $fancybox = $('[data-fancybox]');
    $fancybox.fancybox({
        buttons: [
            'slideShow',
            'fullScreen',
            'thumbs',
            // 'share',
            // 'download',
            // 'zoom',
            'close'
        ],
        lang: 'ru',
        i18n: {
            'ru': {
                CLOSE: 'Закрыть',
                NEXT: 'Вперед',
                PREV: 'Назад',
                ERROR: 'Не удалось установить соединение. <br/> Пожалуйста,
попробуйте позднее.',
                PLAY_START: 'Начать слайдшоу',
                PLAY_STOP: 'Поставить на паузу',
                FULL_SCREEN: 'На полный экран',
                THUMBS: 'Превьюшки',
                DOWNLOAD: 'Скачать',
                SHARE: 'Поделиться',
                ZOOM: 'Увеличить'
            }
        }
    })
})(jQuery);
// end fancybox


// start main
// end main


// start google maps
function googleMapsLoaderCallback() {
    var mapBlock = document.getElementById('contact-map');
    var latLng = {
        lat: 60.077741,
        lng: 30.341899
    };
```

```
    var mapSetting = {
        center: latLng,
        disableDefaultUI: true,
        draggable: false,
        scrollwheel: false,
        zoom: 15
    };
    var dark = {
        styles: [{
            featureType: "all",
            elementType: "all",
            stylers: [{
                invert_lightness: false
            }, {
                saturation: -100
            }, {
                lightness: 20
            }, {
                gamma: 2
            }]
        }],
        name: {
            name: "Dark"
        }
    };

    var styledMapDarkLight = new google.maps.StyledMapType(dark.styles,
dark.name);
    var map = new google.maps.Map(mapBlock, mapSetting);
    map.mapTypes.set('Dark', styledMapDarkLight);
    map.setMapTypeId('Dark');
    var infoWindowContent = '<div>' +
        '<p class="font-weight-bold">город Санкт-Петербург</p>' +
        '<p>ул. Федора Абрамова, д.16/1, кв. 16</p>' +
        '</div>';
    var infoWindow = new google.maps.InfoWindow({
        content: infoWindowContent
    });


    var icon = {
        anchor: new google.maps.Point(384 / 2, 512),
        fillOpacity: .7,
        path: "M192 96c-52.935 0-96 43.065-96 96s43.065 96 96 96 96-43.065
96-96-43.065-96-96-96zm0 160c-35.29 0-64-28.71-64-64s28.71-64 64-64 64 28.71
64 64-28.71 64-64 64zm0-256C85.961 0 0 85.961 0 192c0 77.413 26.97 99.031
172.268 309.67 9.534 13.772 29.929 13.774 39.465 0C357.03 291.031 384 269.413
384 192 384 85.961 298.039 0 192 0zm0 473.931C52.705 272.488 32 256.494 32
192c0-42.738 16.643-82.917 46.863-113.137S149.262 32 192 32s82.917 16.643
113.137 46.863S352 149.262 352 192c0 64.49-20.692 80.47-160 281.931z",
        scale: .15,
        scaledSize: new google.maps.Size(384, 512)
    };

    var marker = new google.maps.Marker({
        icon: icon,
        map: map,
        optimized: false,
        position: latLng
    });
    marker.addListener('click', function () {
        infoWindow.open(map, marker);
    });
```

```
}

// end google maps


// purshedProduct
(function ($) {
    var $apiForm = $('form.api-form');
    var locationSearch = '&' + window.location.search.slice(1);
    var locationPathname = window.location.pathname;
    var $apiPaste = $(".api-paste__table");

    // switch (locationPathname) {
    //     case '/employee/':
    //         var ajax = $.ajax({
    //             dataType: 'json',
    //             url: "../api/purshed-product.php?action=get-employee"
    //         });
    //
    //         ajax.done(function (result) {
    //             if (!result.status) {
    //                 return false;
    //             }
    //
    //             $apiPaste.html(result.html);
    //         });
    //         break;
    //     case 'value2':
    //         break;
    // }

    $apiForm.on('submit', function () {
        var $form = $(this).closest('form');
        var requestUrl = $form.attr('action');
        var ajax = $.ajax({
            data: $form.serialize(),
            dataType: 'json',
            type: 'POST',
            url: requestUrl + locationSearch
        });

        ajax.done(function (result) {
            if (!result.status) {
                return false;
            }

            switch (result.action) {
                case 'sign-in':
                    window.location.replace(result.redirect);
                    break;
                case 'value2':
                    break;
            }
        });
        return false;
    });


})(jQuery);
// purshed-product
```

Скрипт аудентификации Autentification

```
<?php
defined("_ACCESS_INCLUDE") || die("Ошибка доступа");
include_once _ROOT . "/include/sql.php";
session_start();

$action = $_GET["action"];
if ($action === "sign-out") {
    unset($_SESSION["id"]);
    unset($_SESSION["login"]);
    unset($_SESSION["role"]);
    header("Location: /");
    die("redirect");
}

if (!isset($_SESSION["id"]) && _URL_PAGE !== "/sign-in/") {
    header("Location: /sign-in/?page=" . urlencode(_URL_PAGE));
} elseif (isset($_SESSION["id"])) {
    if (_URL_PAGE === "/sign-in/") {
        header("Location: /");
        die("redirect");
    }
    $sql = "SELECT
                `pages`.`id` AS `id`,
                `pages`.`url`,
                `role`.`name`
            FROM
                `pages`
            INNER JOIN `pagesRole` ON `pagesRole`.`idPages` = `pages`.`id`
            INNER JOIN `role` ON `role`.`id` = `pagesRole`.`idRole`
            WHERE
                `pages`.`url` = '" . _URL_PAGE . "' AND `role`.`name` =
'{$_SESSION["role"]}'";
    $mysqli = new sql();
    $result = $mysqli->sqlQuery($sql);

    if (!$row = $result->fetch_assoc()) {
        header("Location: /access-error/");
        die("redirect");
    };
}
?>
```

Все константы вынесены в отдельный файл

```
<?php
define("_ROOT", $_SERVER["DOCUMENT_ROOT"]);
define("_DB_HOST", "localhost");
define("_DB_USER", "voronily_rgrtu");
define("_DB_PASSWORD", "rgrtu4041");
define("_DB_BASE", "voronily_rgrtu");
define("_URL_PAGE", parse_url($_SERVER["REQUEST_URI"], PHP_URL_PATH));
?>
```

Подключение к БД

```
<?php
defined("_ACCESS_INCLUDE") || die("Ошибка доступа");

class sql
{
```

```
    protected $mysqli;

    public function sqlQuery($sql)
    {
        if (!isset($this->mysqli)) {
            $this->connect();
        }
        if (!$result = $this->mysqli->query($sql)) {
            $resultArr = [
                "status" => false,
                "description" => "<strong>Ошибка!</strong> {$this->mysqli->error}"
            ];
            die(json_encode($resultArr));
        };
        return $result;
    }

    private function connect()
    {
        $this->mysqli = new mysqli (_DB_HOST, _DB_USER, _DB_PASSWORD, _DB_BASE);
        if ($this->mysqli->connect_errno) {
            $resultArr = [
                "status" => false,
                "description" => "Не удалось подключиться к БД: {$this->mysqli->connect_error} ({$this->mysqli->connect_errno})"
            ];
            die(json_encode($resultArr));
        }
    }
}

?>
```

Основной каркас страницы

```
<?php
define("_ACCESS_INCLUDE", true);
include_once _ROOT . "/include/authenticate.php";
?>
<!DOCTYPE html>
<html lang="ru" xmlns="http://www.w3.org/1999/html">
<?php include_once _ROOT . "/include/head.php"; ?>
<body>
<div class="svg-container d-none">
    <?php echo file_get_contents(_ROOT . "/include/svg-icon.svg"); ?>
</div>
<div id="wrapper">
    <?php include_once _ROOT . "/include/nav.php"; ?>
    <header class="home-header dark px-2 py-5 invisible" data-scrollspy-class="animation-fade">
        <div class="container h-100">
            <div class="row align-items-center h-100">
                <div class="col">
                    <h1 class="fs-res-1 text-center">
                        <?php
                        echo "Добро пожаловать: {$_SESSION["login"]}!";
                        ?></h1>
                    <h3 class="font-italic fs-res-2 text-center mt-4">
                        <?php
                        echo "Ваша роль в системе: {$_SESSION["role"]}";
```

```
                        ?>
                    </h3>
                </div>
            </div>
        </div>
    </header>
    <main>
        <section class="home-focus px-2 py-5">
            <div class="container">
                <div class="row invisible" data-scrollspy-class="animation-
slide-bottom-medium">
                    <div class="col">
                        <h2 class="font-weight-bold text-center">Расскажу
пару слов о себе</h2>
                        <p>Я увлекаюсь уличным направлением WorkOut. Мне
нравится, кататься на коньках и
                            велосипеде, играть в боулинг и бильярд. Также я
очень люблю кофе с маршмеллоу. Уже сейчас я
                            запустил много интересных проектов. Если вы
хотите узнать больше о моей работе,
                            пожалуйста, возьмите телефон и позвоните мне,
либо свяжитесь со мной по электронной
                            почте. </p>
                    </div>
                </div>
            </div>
        </section>
        <section class="home-project px-2 py-5">
            <div class="container">
                <div class="row invisible" data-scrollspy-class="animation-
scale-up">
                    <div class="col">
                        <h2 class="font-weight-bold text-center">Мои
последние проекты</h2>
                    </div>
                </div>
                <div class="row">
                    <div class="col-12 col-sm-6 col-lg invisible" data-
scrollspy-class="animation-scale-up">
                        <div class="img-card my-3">
                            <img src="/img/project/citigold-private-
client.jpg" alt="">
                            <div class="img-card__item_text">
                                <div class="row align-items-center justify-
content-center h-100">
                                    <div class="col-auto">
                                        <h4 class="text-center text-
white">CPC</h4>
                                        <p class="mt-3 mb-0"><a class="btn
btn-outline-light-1" target="_blank"
href="https://www.citigoldprivateclient.ru">Просмотреть</a>
                                        </p>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                    <div class="col-12 col-sm-6 col-lg invisible" data-
scrollspy-class="animation-scale-up">
                        <div class="img-card my-3">
                            <img src="/img/project/citipriority.jpg" alt="">
                            <div class="img-card__item_text">
```

```
                                    <div class="row align-items-center justify-
content-center h-100">
                                        <div class="col-auto">
                                            <h4 class="text-center text-
white">Citi Priority</h4>
                                            <p class="mt-3 mb-0"><a class="btn
btn-outline-light-1" target="_blank"
href="https://www.citibank.ru/russia/citipriority/rus/form.htm">Просмотреть</
a>
                                            </p>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div class="col-12 col-sm-6 col-lg invisible" data-
scrollspy-class="animation-scale-up">
                                <div class="img-card my-3">
                                    <img src="/img/project/citigold.jpg" alt="">
                                    <div class="img-card__item_text">
                                        <div class="row align-items-center justify-
content-center h-100">
                                            <div class="col-auto">
                                                <h4 class="text-center text-
white">Citi Gold</h4>
                                                <p class="mt-3 mb-0"><a class="btn
btn-outline-light-1" target="_blank"
href="https://www.citigold.ru">Просмотреть</a></p>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div class="col-12 col-sm-6 col-lg invisible" data-
scrollspy-class="animation-scale-up">
                                <div class="img-card my-3">
                                    <img src="/img/project/invest.jpg" alt="">
                                    <div class="img-card__item_text">
                                        <div class="row align-items-center justify-
content-center h-100">
                                            <div class="col-auto">
                                                <h4 class="text-center text-
white">Investments</h4>
                                                <p class="mt-3 mb-0"><a class="btn
btn-outline-light-1" target="_blank"
href="https://www.citibank.ru/russia/invest/rus/invest.htm">Просмотреть</a>
                                                </p>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </section>
                <section class="home-life px-2 py-5">
                    <div class="container">
                        <div class="row invisible" data-scrollspy-class="animation-
scale-up">
                            <div class="col">
```

```
                        <h2 class="font-weight-bold text-center">Какой я в
обычной жизни</h2>
                    </div>
                </div>
                <div class="mt-2 no-gutters row">
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/01-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/01-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/02-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/02-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/03-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/03-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/04-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/04-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/05-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/05-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/06-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/06-
life-thumbnail.jpg" alt="">
                            </a>
```

```
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/07-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/07-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/08-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/08-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/09-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/09-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/10-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/10-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/11-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/11-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                    <div class="col-6 col-md-3 invisible" data-scrollspy-
class="animation-scale-up">
                        <div class="m-1">
                            <a data-fancybox="home-life" href="/img/life/12-
life-large.jpg">
                                <img class="img-fluid" src="/img/life/12-
life-thumbnail.jpg" alt="">
                            </a>
                        </div>
                    </div>
                </div>
```

```
            </div>
        </section>
    </main>
    <?php include_once _ROOT . "/include/footer.php"; ?>
</div>
<?php include_once _ROOT . "/include/scripts.php"; ?>
</body>
</html>
```