РЕФЕРАТ

Выпускная квалификационная работа содержит пояснительную записку на 67 странице формата А4, включающую 5 разделов, 17 рисунков, 4 таблицы, 8 источников, 1 приложение.

ЭКСТРЕННЫЕ СЛУЖБЫ, ВЫЗОВ, СИСТЕМА, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КАРТА, СКОРОСТЬ.

Объектом исследования в рамках данной работы является система для вызова экстренных служб.

Целью данной̆ работы является проектирование архитектуры иразработка программного обеспечения, вызывающего экстренные службы по запросу пользователя.

В результате выполнения данной работы проведено исследование существующей системы вызовов ЭС, существующих API для работы с геолокацией пользователя и Framework для разработки мобильных приложений, выявлены их достоинства и недостатки. Разработана архитектура и программное обеспечение новой системы.

Содержание

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 8](#_Toc168653575)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 9](#_Toc168653576)

[ВВЕДЕНИЕ 10](#_Toc168653577)

[1. Исследовательский раздел 12](#_Toc168653578)

[1.1. Описание существующей системы вызова пожарных служб 12](#_Toc168653579)

[1.2. Система распределения телефонных звонков по оперативным службам экстренного реагирования Российской Федерации «Система – 112» 13](#_Toc168653580)

[1.3. Функция Crash Detection 14](#_Toc168653581)

[1.4. Сбор статистики 14](#_Toc168653582)

[1.5. Анализ архитектуры разработки мобильных приложений 15](#_Toc168653583)

[2. Конструкторский раздел 16](#_Toc168653584)

[2.1. Выбор средств разработки 16](#_Toc168653585)

[2.2. Проектирование бизнес-процессов 17](#_Toc168653586)

[2.3. Проектирование клиент-серверного взаимодействия 21](#_Toc168653587)

[2.4. Проектирование системы разграничения доступа к информации 22](#_Toc168653588)

[2.5.1. Сценарий входа в приложение 25](#_Toc168653589)

[2.5.2. Интерфейс вызова пожарной бригады 26](#_Toc168653590)

[2.5.3. Интерфейс бокового меню и списка сохраненных адресов 31](#_Toc168653591)

[2.6. Проектирование базы данных 33](#_Toc168653592)

[2.7. Архитектура мобильного приложения 37](#_Toc168653593)

[2.8. Вывод по разделу 39](#_Toc168653594)

[3. Разработка 40](#_Toc168653595)

[3.1. Разработка базы данных 40](#_Toc168653596)

[3.2. Разработка мобильного приложения 48](#_Toc168653597)

[3.2.1. Разработка View элементов 48](#_Toc168653598)

[3.2.2. Разработка Presenter для хранения и обработки данных 50](#_Toc168653599)

[3.2.3. Разработка Interactor для организации запросов к серверу 52](#_Toc168653600)

[3.2.4. Разработка Entity для хранения данных 54](#_Toc168653601)

[3.3. Вывод по разделу 55](#_Toc168653602)

[4. Социально-экономическая эффективность 56](#_Toc168653603)

[4.1. Социальная эффективность 56](#_Toc168653604)

[4.1.1. Ускорение реагирования на чрезвычайные ситуации 56](#_Toc168653605)

[4.1.2. Повышение безопасности населения 56](#_Toc168653606)

[4.1.3. Инклюзивность 56](#_Toc168653607)

[4.2. Экономическая эффективность 57](#_Toc168653608)

[4.2.1. Снижение затрат на экстренные службы 57](#_Toc168653609)

[4.2.2. Косвенные экономические выгоды 57](#_Toc168653610)

[4.3. Определение капитальных вложений 57](#_Toc168653611)

[4.4. Расчет экономического эффекта 58](#_Toc168653612)

[4.5. Вывод по разделу 60](#_Toc168653613)

[5. Информационная безопасность 61](#_Toc168653614)

[5.1. Вывод по разделу 62](#_Toc168653615)

[ЗАКЛЮЧЕНИЕ 63](#_Toc168653616)

[ПРИЛОЖЕНИЕ А 65](#_Toc168653617)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

API – интерфейс взаимодействия двух компьютерных программ.

Framework – программная платформа, которая упрощает создание и поддержку технически сложных или нагруженных проектов.

Desktop - приложение – приложение устанавливаемое на домашний компьютер.

Мобильное приложение – приложение, устанавливаемое на мобильные (смартфоны, планшеты т.д.) устройства.

База данных — это упорядоченный набор структурированных данных, которые хранятся в электронном виде в компьютерной системе.

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ЭС – экстренные службы.

ПК – персональный компьютер.

БД – база данных.

ПО – программное обеспечение.

МЧС - министерство РФ по делам гражданской обороны, чрезвычайным ситуациям и ликвидации последствий стихийных бедствий.

# ВВЕДЕНИЕ

В настоящее время вызов экстренных служб выполняется через звонок по закрепленному номеру 112, что не всегда удобно и не всегда реализуемо:

* Фактор стресса приводит к невозможности вспомнить номер и правильно его ввести;
* Отсутствие отслеживания статуса вызова служб;
* Сложность указания адреса, если вызывающий не знает где находится.

Одним из методов устранения вышеуказанных недостатков является разработка системы вызова экстренных служб с помощью мобильного приложения.

Актуальность проектирования и разработки такой системы вызова экстренных служб, подтверждается опросом среди жителей Пермского Края.

Полученные результаты:

* 13% смогли вспомнить номера всех экстренных служб;
* 87% опрошенных, высказали мнение о необходимости иметь мобильное приложение на своём смартфоне.

Целью выпускной квалификационной работы «Разработка программного обеспечения автоматизированной системы обработки данных, учёта, оповещения о чрезвычайных ситуациях» является проектирование архитектуры и создание кроссплатформенного программного обеспечения для вызова экстренной службы – «Государственная противопожарная служба Российской Федерации» (далее ГПС).

Для достижения целей в рамках выпускной квалификационной работы поставлены и решены следующие задачи:

* Анализ существующих систем вызова экстренных служб;
* Проектирование архитектуры пользовательского взаимодействия: разделение по ролям, связи между ними и доступность функционала системы, для каждой из ролей.
* Разработка программного обеспечения системы оповещения МЧС о чрезвычайных ситуациях для мобильных устройств.
* Анализ и выбор структуризации программного кода, для грамотного разделения на модули и возможности дальнейшего развития ПО.
* Анализ и выбор платформы для ускорения разработки приложений (далее Framework)
* Разработка интерфейсов мобильного приложения

# Исследовательский раздел

## Описание существующей системы вызова пожарных служб

В настоящее время используется система вызова пожарных служб, которая не изменялась с 1930-х годов.

Алгоритм процессов вызова и реагирования ГПС:

* + - 1. Вызов по номеру 101 или 112 (единый номер вызова экстренных служб);
      2. Операторы диспетчерского центра принимают вызов, обрабатывают запрос и координируют действия бригады;
      3. Пожарная бригада выезжает на место происшествия в соответствии с указаниями диспетчера.

Алгоритм имеет следующие недостатки:

1. Влияние фактора стресса приводить к невозможности вызова пожарной службы по номеру телефона.
2. Диспетчер вручную записывает информацию о месте возгорания, ФИО звонящего, особенностях возгорания, что не исключает ошибок в написании.
3. Плохая связь так же приводит к голосовым помехам, что усложняет определение слов говорящего.
4. Кроме того, адрес возгорания не всегда может быть определён точно.

## Система распределения телефонных звонков по оперативным службам экстренного реагирования Российской Федерации «Система – 112»

Цели работы «Системы – 112»:

* Организация вызова экстренных оперативных служб по принципу «одного окна»;
* Организация комплекса мер, обеспечивающих ускорение реагирования и улучшение взаимодействия экстренных оперативных служб при вызовах (сообщениях о происшествиях);
* Разработка способов вызова экстренных служб в соответствии с единым Европейским стандартом.

«Система – 112» уже работает в автоматическом режиме со стационарными устройствами на контролируемых участках и автомобильных терминалах, оборудованных системой реагирования на аварии «ЭРА-ГЛОНАСС» и терминалах ГЛОНАСС/GPS, установленных на транспортных средствах экстренных оперативных служб, привлеченных к реагированию на происшествие, и транспортных средствах, перевозящих опасные грузы.

У данной системы имеется недостаток – отсутствие возможности граждан использовать её в автоматическом режиме (автоматическое указание данных о пользователе: личные данные, точное место вызова и т.д.), с помощью мобильных устройств. Решением этой проблемы является разработка мобильного приложения для автоматического вызова экстренных служб

## Функция Crash Detection

Crash Detection - система, разработанная компанией Apple для IPhone версий 14 и выше, а также для Apple Watch SE/Ultra/Series 8. Данная система работает на основе технологий:

1. Гироскопа и акселерометра с высоким ускорением;
2. GPS;
3. Барометра;
4. Микрофона;

Когда водитель находится в движении, устройство начинает принимать информацию с датчиков. Если функция обнаружит признаки дорожно-транспортного происшествия (ДТП), на экране Apple Watch или ‌iPhone‌ в течение 10 секунд отобразится предупреждение. Если в течении 10 секунд пользователь не взаимодействовал с устройством, то начнется обратный отсчет до автоматического вызова экстренных служб.

В состав Crash Detection входит таймер для отмены вызова, и реализована автоматическая рассылка избранным контактам сообщения о пожаре.

## Сбор статистики

«Система - 112» использует данные от мобильных операторов, что не гарантирует точную информацию о месте возгорания и лишает возможности автоматического внесения информации о происшествии в базу данных. Использование автоматической системы устраняет эти проблемы.

Сбор статистики требуется для определения точного места возгорания, автоматического внесения данных о месте происшествия и аналитика данных, с дальнейшим выявлением потенциально опасных сооружений гражданской инфраструктуры.

Мобильное приложение автоматизирует процесс сбора и передачи, данный в БД.

## Анализ архитектуры разработки мобильных приложений

Проведен анализ вариантов архитектуры мобильных приложений: MVC, MVVM и VIPER. Основными критериями выбора архитектуры являются: отсутствие пересечений модулей, изменение модулей без изменений кода других модулей, отсутствие ограничений в переключении между экранами, высокая скорость тестирования, читаемость кода при больших размерах проекта. Недостатки и достоинства указаны в (таблице 1).

Таблица – 1 – достоинства и недостатки архитектур MVVM, MVC, VIPER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Отсутствие пересечений модулей | Изменение модулей без изменений кода других модулей | Отсутствие ограничений в переключении между экранами | Высокая скорость тестирования | Читаемость кода при больших размерах проекта |
| MVVM | + | + | - | + | - |
| MVC | - | + | - | + | - |
| VIPER | + | + | + | + | + |

В результате проведенного анализа выбрана архитектура VIPER. Основным аргументом выбора в пользу архитектуры VIPER является возможность увеличение размера приложения, что сложно реализовать при использовании архитектур MVC и MVVM из-за кратного увеличения времени работы с кодом.

# Конструкторский раздел

## Выбор средств разработки

Для разработки серверной части приложения выбран язык программирования C#. Данное решение обусловлено использованием одного языка для серверной и клиентской частей, так как разработка мобильного приложения с использованием .net MAUI поддерживает только язык программирования C#.

В качестве системы управления базами данных выбран PostgreSQL.

Для реализации клиентской части приложения выбраны:

* Язык XAML для разметки View элементов приложения;
* Язык программирования C# для программирования логики работы пользовательского интерфейса и бизнес-логики сервера;
* Фреймворк .net MAUI для кроссплатформенности приложения.

## Проектирование бизнес-процессов

В рамках ВКР разработана модель основного бизнес-процесса на мобильных устройствах в виде диаграммы деятельности вызова экстренных служб на рисунке 2.2.1.

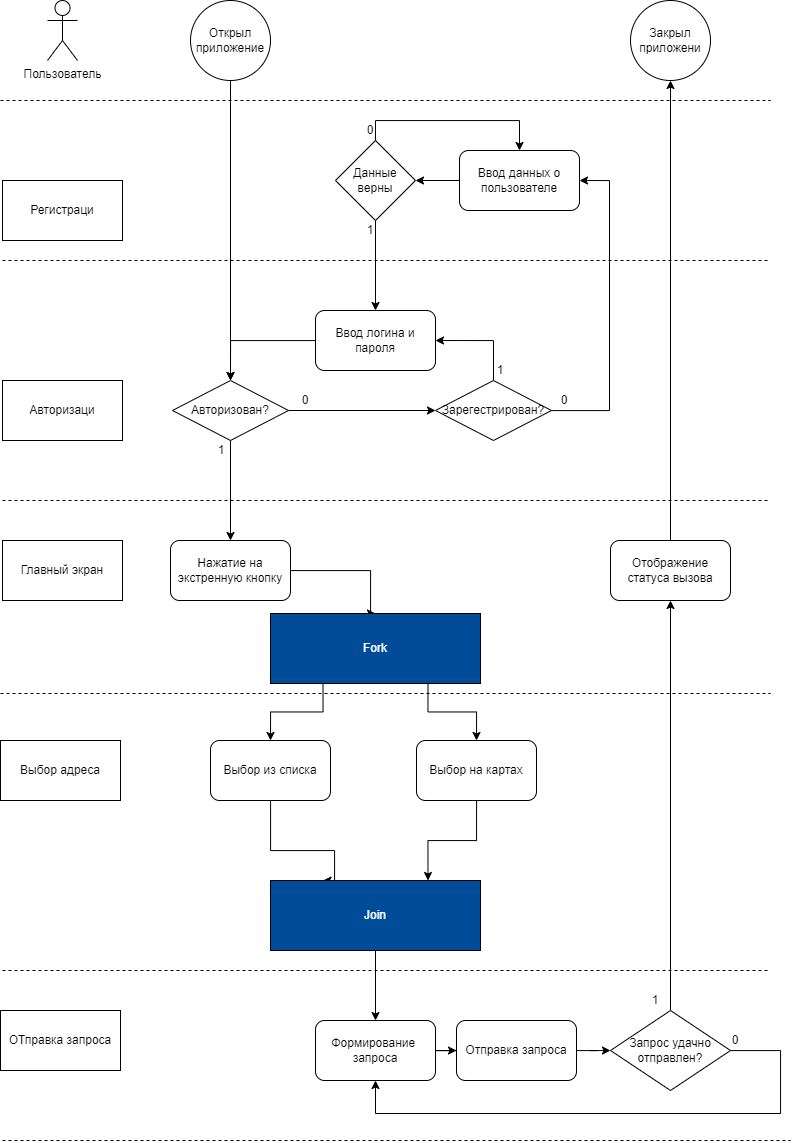


Рисунок 2.2.1. – Модель бизнес – процесса отправки заявки пользователем

При проектировании бизнес-процесса учтены, различные состояния аккаунта пользователя: авторизован, зарегистрирован и отсутствует аккаунт.

При проектировании обработки данных на сервере, на рисунке 2.2.2, учтены возможные потери при передаче пакетов данных, чтобы при их не удачной обработке, оператор мог просмотреть и среагировать на заявку.

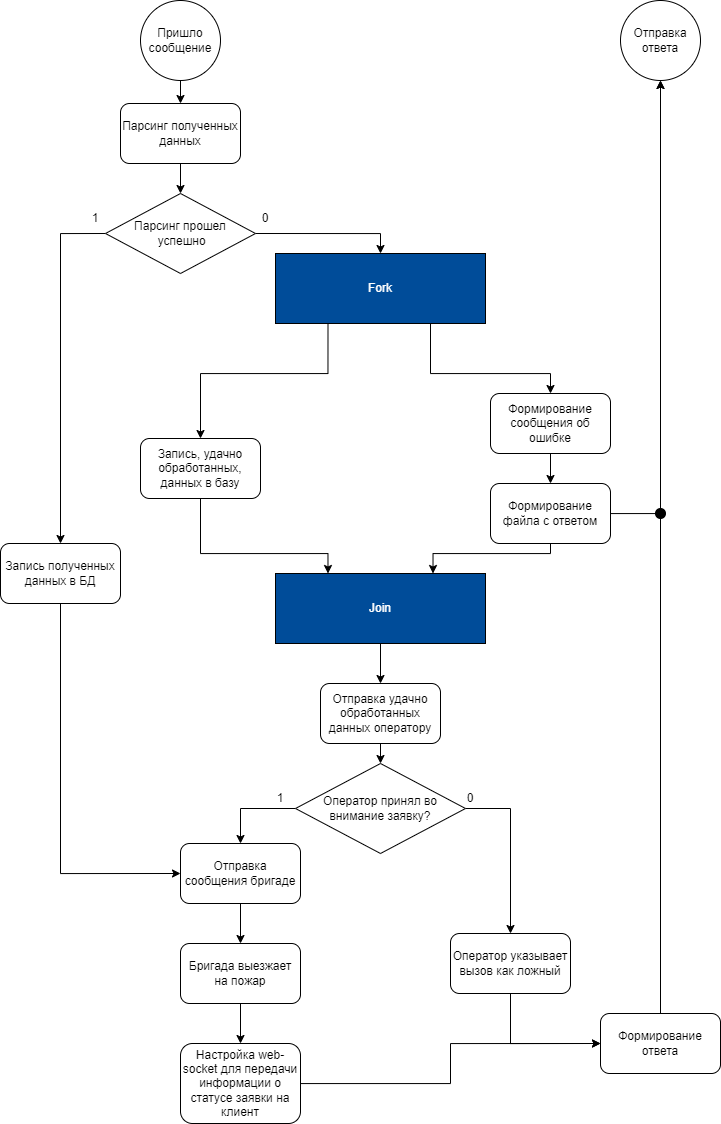


Рисунок 2.2.2 – Обработка заявки на сервере

При проектировании работы роли «Оператор» разработан бизнес-процесс обработки заявок оператором, рисунок 2.2.3.

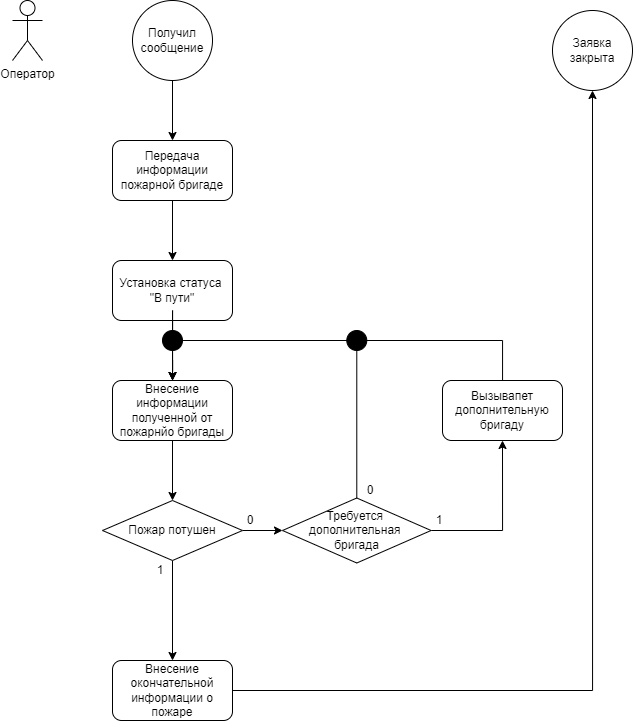


Рисунок 2.2.3 – Бизнес-процесс принятия заявки оператором

В процессе проектирования учтены ситуации не хватки пожарной бригады. Так же пожарная бригада может передавать информацию оператору, для внесения дополнительной информации о возгорании.

Бизнес-процесс работы инспектора, рисунок 2.2.4, включает в себя 3 основных действия, установка статуса заявки «На проверке», проверка объекта гражданской инфраструктуры, установка статуса «Проверка пройдена», при положительном решении инспектора. Все заявки имеют изначальный статус «Проверка не пройдена», то есть при отрицательном решении заявка переходит из статуса «Проверяется» в статус «Проверка не пройдена».

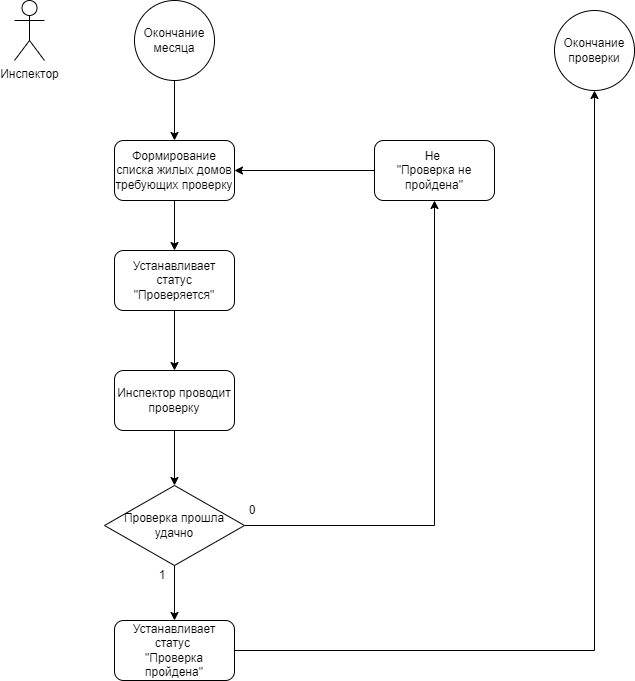


Рисунок 2.2.4 – Бизнес-процесс проверки объектов гражданской инфраструктуры

## Проектирование клиент-серверного взаимодействия

Клиент-серверное взаимодействие будет происходить на базе архитектуры REST API, так как передача данных будет выполняться в формате JSON. Протоколом передачи данных будет являться HTTPS для сохранности данных пользователя с использованием SSL сертификатам.

Формат URL имеет вид <https://domen/catalog/resource> и состоит из четырех компонентов:

* https:// - используемый протокол передачи данных;
* domen/ - доменное имя, идентифицирующее разработанный ресурс в сети;
* catalog/ - путь к каталогу, в котором хранятся ресурсы системы;
* resource - ресурс системы, обрабатывающий запрос.

На рисунке 2.3.1. показано схематическое описание работы REST API.

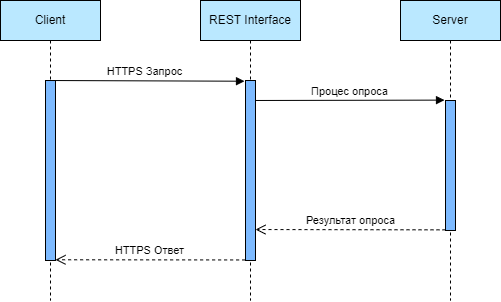


Рисунок 2.3.1. – Диаграмма REST API

При работе API используются 3 вида запросов:

* GET для получения данных с сервера, например, для получения сохранённых адресов пользователем ;
* POST для создания новых записей, например, для регистрации пользователей или сохранения нового адреса;
* DELETE для удаления записей, например, пользователей или сохраненных адресов;

В заголовке запроса предается токен авторизации, для проверки пользователя на реальность. В теле запроса передается JSON-файл со значениями полей для обмена данными между клиентом и сервером. Графически общий вид запроса представлен на рисунке 2.3.2.



Рисунок 2.3.2. – Общий вид запроса

## Проектирование системы разграничения доступа к информации

Для разграничения функционала информационных и функциональных возможностей системы разработаны сценарии ролей доступа пользователей к автоматической системе оповещения МЧС о чрезвычайных ситуациях, представленные в таблице 2.

Таблица 2 – Роли доступа

|  |  |
| --- | --- |
| Роль | Описание роли |
| Гражданин | Пользователь, который может изменять свои данные: сохраненные адреса, ФИО, номер телефона, пароль, логин |

Продолжение таблицы 2

|  |  |
| --- | --- |
| Администратор | Может управлять всеми пользователями, вносить изменения в их данные, просматривать статистику вызовов |
| Контроллер | Может просматривать статистику, по вызовам и информацию по плановым проверкам |
| Оператор | Принимать вызовы и добавлять информацию о вызове, поступающую от бригад, во время пожаротушения. |

На рисунке 2.7.1 изображена диаграмма вариантов использования системы, в зависимости от роли пользователя.

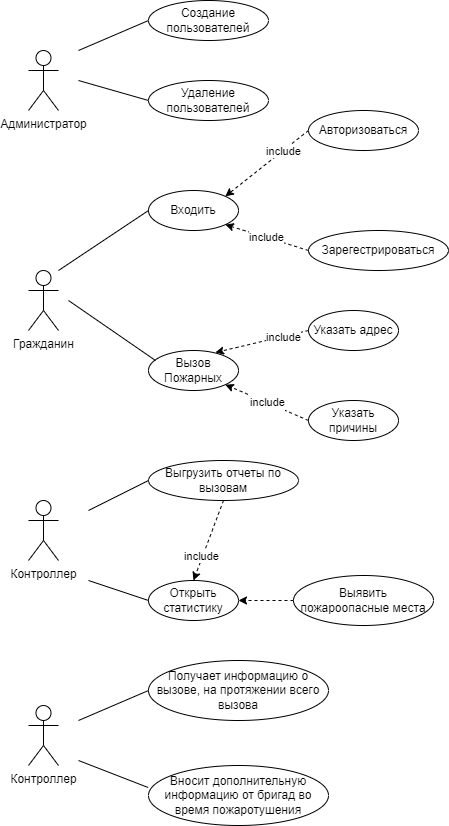


Рисунок 2.7.1 – Диаграмма вариантов использования системы доступа

* 1. **Проектирование пользовательского интерфейса мобильного приложения**

Для взаимодействия пользователя с мобильным приложением разработан интерфейс, состоящий из девяти оконных форм, доступ к которым имеет пользователь с ролью «Гражданин» (таблица 2.3).

### Сценарий входа в приложение

Сценарий входа в приложение содержит два окна: окно загрузки приложения (рисунок 2.5.1) и окно авторизации пользователя (рисунок 2.5.2).

Окно загрузки отображается при запуске приложения, где происходит авторизация по паролю и логину, сохраненным на телефоне, в соответствии с данными последней авторизации.



Рисунок 2.5.1 – Окно загрузки

Окно авторизации отображается в двух случаях: при отсутствии файла с сохраненным паролем и логином или при неудачной авторизации по сохраненным логину и паролю.

Элементы интерфейса содержат:

* поле ввода логина;
* поле ввода пароля;
* кнопка «Войти»;
* кнопка «Зарегистрироваться»;
* кнопка «Вызвать звонком», для оповещения МЧС по звонку, когда ситуация имеет экстренный характер, а автоматическая авторизация не произошла или при отсутствии сети интернет.

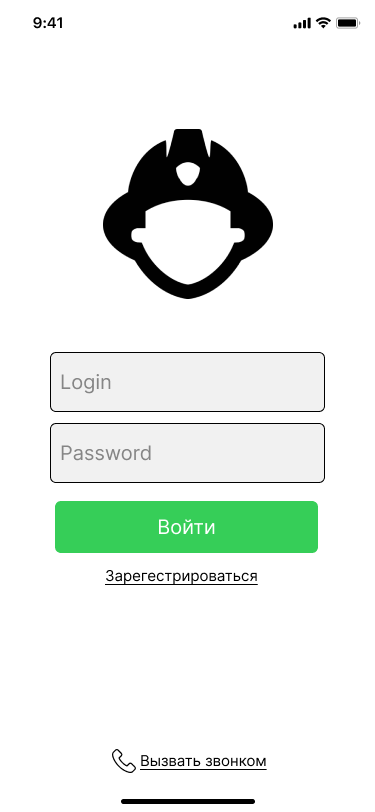


Рисунок 2.5.2 – Окно авторизации

### Интерфейс вызова пожарной бригады

Интерфейс сценария вызова пожарной бригады имеет три окна: для выбора адреса, для ввода комментария, окно с кнопкой «Вызов» для вызова экстренной службы.

Для выбора адреса вызова в оконной форме расположена карта с указанием геолокации пользователя и списком важных для конкретного пользователя адресов (рисунок 2.5.3).

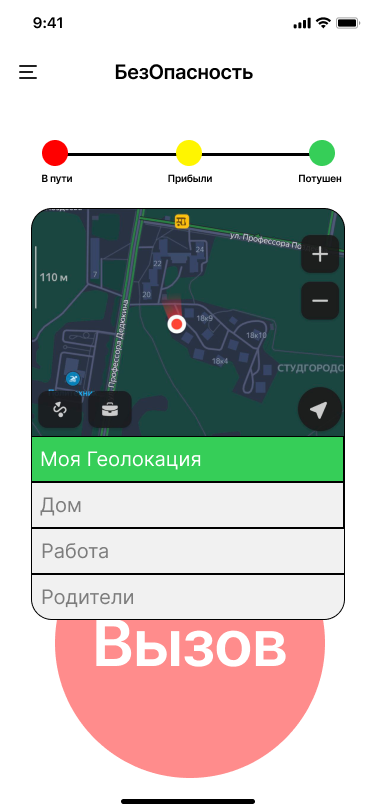


Рисунок 2.5.3 – Главный экран приложения

После выбора адреса скрывается всплывающий список с важными адресами, и отображается поле комментария для уточнения подробностей о чрезвычайной ситуации (рисунок 2.5.4).

Добавление комментария является опциональной функцией поэтому кнопка «Добавить комментарий» активна всегда, и при нажатии на неё, появляется возможность вызова пожарной бригады (рисунок 2.5.5).

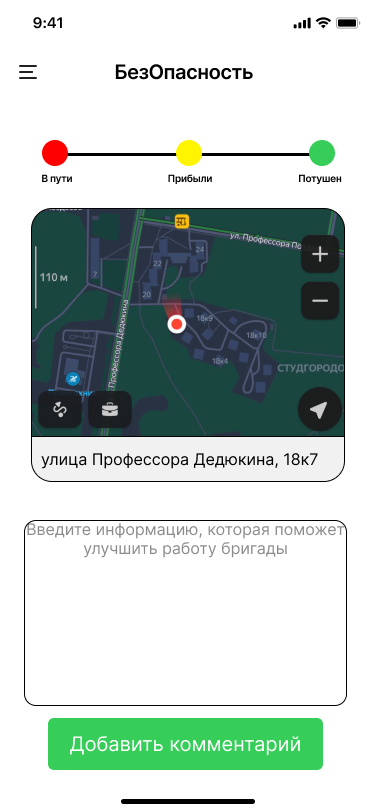


Рисунок 2.5.4 – Главный экран после выбора адреса

Для отправки пользователем заявки экстренным службам добавлена кнопка «Вызов».

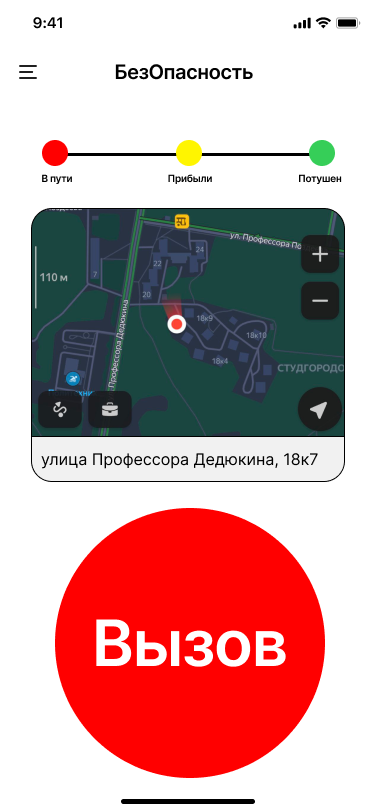


Рисунок 2.5.5 – Главный экран с кнопкой вызова

В случае корректной обработки запроса сервером, над картой отображается иконка с автомобилем экстренной службы, демонстрирующая статус вызова (рисунок 2.5.6).

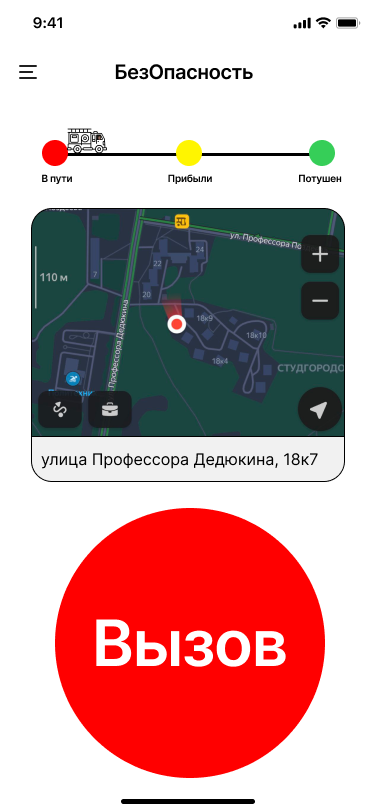


Рисунок 2.5.6 – Главный экран после вызова пожарной бригады

### Интерфейс бокового меню и списка сохраненных адресов

Для отображения информации о пользователе, разработан интерфейс бокового меню (рисунок 2.5.7), на котором отображаются: ФИО и домашний адрес. Ниже расположена кнопка выхода из аккаунта и кнопка для перехода в список важных для пользователя адресов (рисунок 2.5.8).

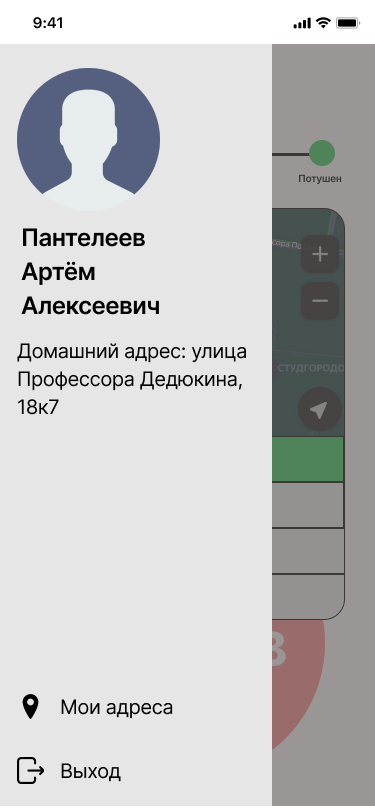


Рисунок 2.5.7 – Боковое меню

Для просмотра списка важных адресов пользователей разработан tableView-интерфейс, в котором в виде таблице отображаются адреса и их псевдонимы. Например, пос. Кез, улица Энергетиков, 17 – «Родители»

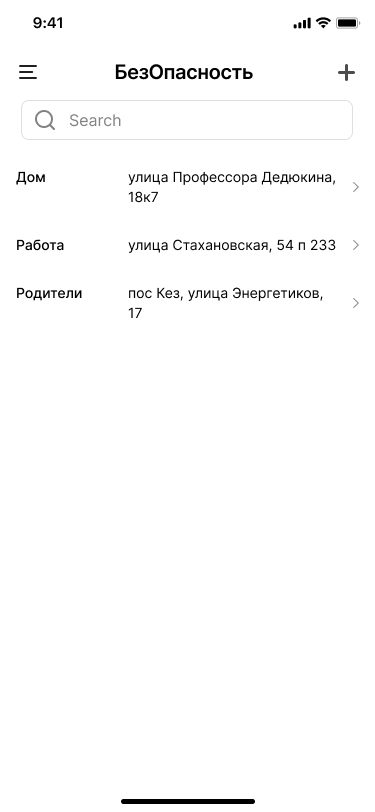


Рисунок 2.5.8 – Список сохраненных адресов

## Проектирование базы данных

Модель базы данных в виде ER-диаграммы, разработана в нотации Джеймса Мартина представлена на рисунке 2.5.1.

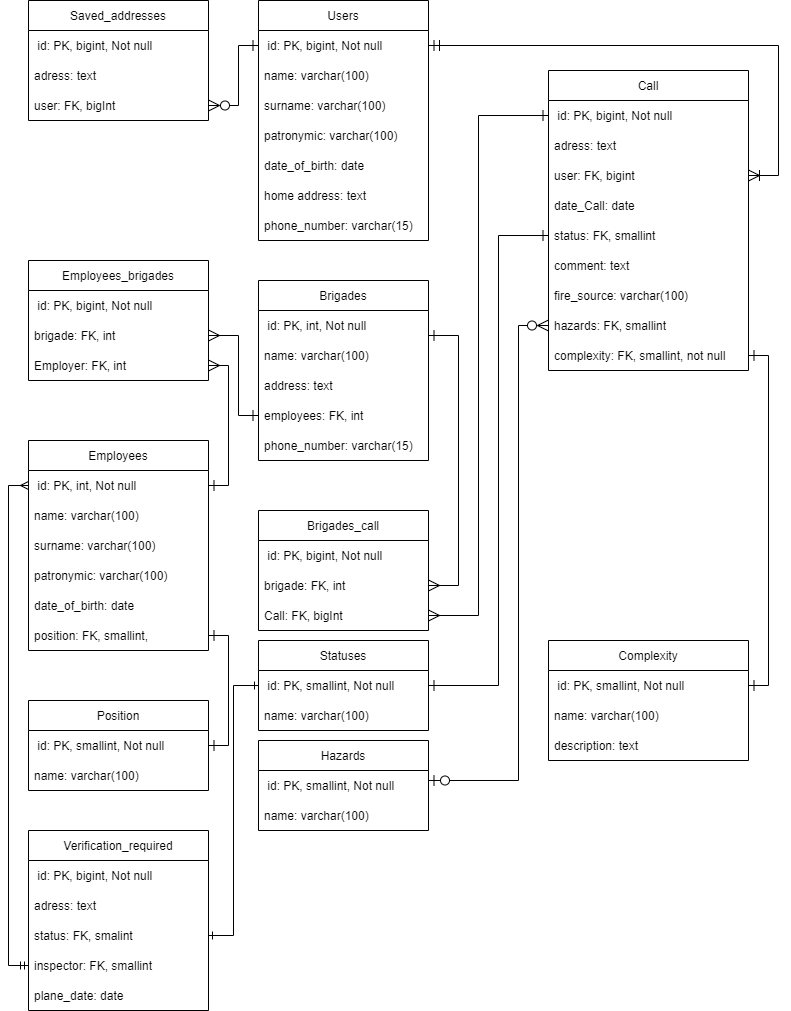


Рисунок 2.5.1 – Физическая модель базы данных

В таблице 3 представлено описание сущностей и полей в модели базы данных.

Таблица 3 – Описание сущностей БД

|  |  |
| --- | --- |
| Сущность | Описание полей |
| Saved\_adress  Сохраненные адреса | id – Уникальный идентификатор записи  address – Сохраненный пользователем адрес для выбора из всплывающего списка  user – Уникальный идентификатор пользователя, сохранившего адрес для его выбора из списка  login – логин пользователя  password – пароль пользователя, записанный в хешированном виде |
| Employees  Сотрудники пожарной службы | id – Уникальный идентификатор сотрудника  name – Имя сотрудника  surname – Фамилия сотрудника  patronymic – Отчество сотрудника  date\_of\_birth – Дата рождения сотрудника  position – Должность сотрудника |
| Position  Должности | id – Уникальный идентификатор должности сотрудника  name – Название должности |

Продолжение таблицы 3

|  |  |
| --- | --- |
| Сущность | Описание полей |
| Verification\_required  Адреса требующие проверку пожарными службами | id – Уникальный идентификатор проверки  address– Адрес, который требуется проверить  status – Статус проверки  inspector – Сотрудник ответственный за проверку  plane\_date – плановая дата проверки |
| Users  Пользователи | id – Уникальный идентификатор пользователя  name – Имя пользователя  surname – Фамилия пользователя  patronymic – Отчество пользователя  date\_of\_birth – Дата рождения пользователя  home\_address – Домашний адрес пользователя  phone\_number – Номер телефона пользователя |
| Brigades  Бригады пожарных | id – Уникальный идентификатор пожарной бригады  name – Название пожарной бригады  address – Адрес пожарной бригады  phone\_number – Номер телефона пожарной бригады |

Продолжение таблицы 3

|  |  |
| --- | --- |
| Сущность | Описание полей |
| Statuses  Статусы | id – Уникальный идентификатор статуса  name – Название статуса |
| Hazards  Опасные факторы | id – Уникальный идентификатор фактора опасности  name – Название фактора опасности |
| Complexity  Сложность пожара | id – Уникальный идентификатор сложности пожара  name – название сложности пожара  description – описание сложности пожара |
| Employees\_brigades  Связующая таблица сотрудников и бригад | id – уникальный идентификатор записи  brigade\_id – уникальный идентификатор бригады  employer\_id – уникальный идентификатор сотрудника |
| Brigades\_call  Связующая таблица бригад и вызовов | id – уникальный идентификатор записи  brigade\_id – уникальный идентификатор бригады  call\_id – уникальный идентификатор вызова |

### Архитектура мобильного приложения

В рамках ВКР приложение разработано с использованием архитектурного паттерна VIPER (View, Interactor, Presenter, Entity, Router), который позволяет разделить ответственность между компонентами (таблица 2.8.1) и обеспечить высокую модульность и тестируемость кода.

Таблица 4 – Описание компонентов архитектуры VIPER

| Компонент | Описание компонента |
| --- | --- |
| View  Вид | Вид отвечает за отображение данных и взаимодействие с пользователем. Он получает информацию от Presenter и отображает ее, а также передает действия пользователя обратно Presenter.  Например, компонент отображающий окно Главного экрана. |
| Interactor  Интерактор | Интерактор содержит бизнес-логику приложения. Он отвечает за обработку данных, выполнение запросов к API и взаимодействие с Entity.  Например, класс отвечающий за создание запросов к серверу и обработки ответов. |
| Presenter  Презентер | Презентер выступает в роли посредника между View и Interactor. Он получает данные от Interactor и подготавливает их для отображения во View. Презентер также получает действия пользователя от View и передает соответствующие команды Interactor. |
| Entity  Сущность | Сущности представляют собой модели данных, используемые в приложении. Они определяют структуру данных, которые будут использоваться Interactor и Presenter.  Например, структуры для хранения информации о пользователе, адресах и т.д. |

Продолжение таблицы 4

|  |  |
| --- | --- |
| Компонент | Описание компонента |
| Router  Роутер | Роутер отвечает за навигацию в приложении. Он управляет переходами между экранами и передачей данных между модулями VIPER. Роутер не содержит бизнес-логику.  Например модель для переключения между экранами вызова и экраном со списком сохраненных адресов. |

Структура архитектуры графически описана на рисунке 2.8.1.

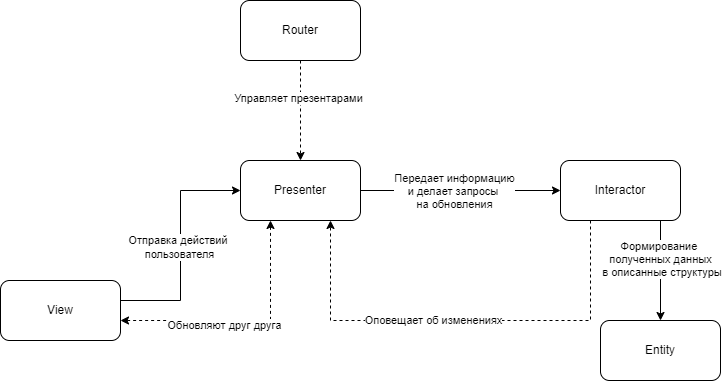


Рисунок 2.7.1 – Схема архитектуры VIPER

Примером использования архитектуры для мобильного приложения является боковое меню (рисунок 2.5.7).

* View отображает ФИО пользователя, адрес и кнопки выхода, перехода на список сохраненных адресов;
* Interactor запрашивает данные о пользователя из базы данных;
* Presenter формирует полученные данные и передает их View для отображения, обрабатывает действия пользователя и вызывает соответствующие методы Interactor;
* Entity определяет модель данных для пользователя, которая используется Presenter и Interactor;
* Router управляет навигацией, например переходом на экран со списком сохраненных адресов.

Таким образом, архитектура VIPER обеспечивает структурированность, модульность и тестируемость приложения, что особенно важно при разработке перспективных мобильных систем.

## Вывод по разделу

В разделе разработаны бизнес-процессы вызова пожарной бригады, работы оператора и проведения проверок инспектором.

Разработаны экранные формы:

* авторизация пользователя;
* главный экран в разных состояниях: выбор места происшествия, добавление комментария, вызов экстренной службы, просмотр статуса вызова;
* боковое меню;
* список важных для пользователя адресов.

Спроектировано разграничение функционала по ролям.

Спроектирована БД системы автоматизированного оповещения МЧС о чрезвычайных ситуациях, диаграмма в нотации Джеймса Мартина рисунок 2.5.1.

Выбраны средства разработки программного обеспечения, и архитектура с учетом возможности дальнейшего развития приложения.

# Разработка

## Разработка базы данных

База данных для системы автоматического оповещения МЧС об экстренных ситуациях реализована с помощью PostgreSQL.

Листинг 1 – Создание таблицы пользователи

CREATE TABLE Users (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

surname VARCHAR(50) NOT NULL,

patronymic VARCHAR(50),

date\_of\_birth DATE NOT NULL,

home\_address VARCHAR(255),

login varchar(30)

password varchar(60)

phone\_number VARCHAR(15)

);

В таблице Users описаны поля с информацией о пользователе:

|  |  |
| --- | --- |
| * «id» - | первичный ключ с автоинкрементом, хранящий уникальный номер записи о пользователе; |
| * «name» - | поле с именем пользователя, состоящее из 50 символов. Не может быть пустым; |
| * «surname» - | поле с фамилией пользователя, состоящее из 50 символов. Не может быть пустым; |
| * «patronymic» - | поле с отчеством пользователя, состоящее из 50 символов. Может быть пустым; |
| * «date\_of\_birth» - | дата рождения пользователя. Не может быть пустым; |
| * «home\_address» - | домашний адрес пользователя. Состоит из 255 символов; |
| * «phone\_number» - | номер телефона пользователя. Состоит из 15 символов. |

Листинг 2 – Создание таблицы с сохраненными адресами

CREATE TABLE Saved\_addresses (

id INT PRIMARY KEY AUTO\_INCREMENT,

address VARCHAR(255) NOT NULL,

user\_id INT,

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

Описание полей начинаются строго друг под другом(без тире)

В таблице Saved\_addresses описаны поля с информацией о сохраненных адресах пользователя:

|  |  |
| --- | --- |
| * «id» - | уникальный идентификатор записи (целое число, автоинкремент); |
| * «address» - | Сохраненный пользователем адрес для выбора из всплывающего списка (строка длиной до 255 символов, обязательное поле); |
| * «user\_id» - | Уникальный идентификатор пользователя, сохранившего адрес для его выбора из списка (целое число). Внешний ключ ссылающий на запись в таблице пользователей. |

Листинг 3 – Создание таблицы с должностями

CREATE TABLE Position (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL

);

В таблице Position описаны поля с информацией о существующих должностях в системе:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор должности сотрудника (целое число, автоинкремент). Первичный ключ; |
| * «Name» - | Название должности (строка длиной до 100 символов, обязательное поле). |

Листинг 4 – Создание таблицы со статусами

CREATE TABLE Statuses (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL

);

В таблице statuses описаны поля с информацией о статусах, существующих в системе:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор статуса (целое число, автоинкремент); |
| * «name» - | Название статуса (строка длиной до 100 символов, обязательное поле). |

Листинг 5 – Создание таблицы с опасными факторами

CREATE TABLE Hazards (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL

);

В таблице Hazards описаны поля с информацией об опасных факторах:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор фактора опасности (целое число, автоинкремент); |
| * «name» - | Название фактора опасности (строка длиной до 100 символов, обязательное поле). |

Листинг 6 – Создание таблицы

CREATE TABLE Complexity (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

description TEXT NOT NULL

);

В таблице сложность описаны поля для хранения списка со сложностями чрезвычайных ситуациях:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор сложности пожара (целое число, автоинкремент); |
| * «name» - | Название сложности пожара (строка длиной до 100 символов, обязательное поле); |
| * «description» - | Описание сложности пожара (текст, обязательное поле). |

Листинг 7 – Создание таблицы сотрудник

CREATE TABLE Employees (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

surname VARCHAR(50) NOT NULL,

patronymic VARCHAR(50),

date\_of\_birth DATE NOT NULL,

position\_id INT,

FOREIGN KEY (position\_id) REFERENCES Position(id)

);

В таблице Employees описаны поля для хранения информации о сотрудниках:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор сотрудника (целое число, автоинкремент); |
| * «name» - | Имя сотрудника (строка длиной до 50 символов, обязательное поле); |
| * «Surname» - | Фамилия сотрудника (строка длиной до 50 символов, обязательное поле); |
| * «patronymic» - | Отчество сотрудника (строка длиной до 50 символов, необязательное поле); |
| * «date\_of\_birth» - | Дата рождения сотрудника (тип DATE, обязательное поле); |
| * «position\_id» - | Должность сотрудника (целое число), внешний ключ, ссылающийся на поле id в таблице Position. |

Листинг 8 – Создание таблицы для записи заявок на проверку

CREATE TABLE Verification\_required (

id INT PRIMARY KEY AUTO\_INCREMENT,

address VARCHAR(255) NOT NULL,

status\_id INT,

inspector\_id INT,

plan\_date DATE NOT NULL,

FOREIGN KEY (status\_id) REFERENCES Statuses(id),

FOREIGN KEY (inspector\_id) REFERENCES Employees(id)

);

В таблице Verification\_required описаны поля с информацией о созданных заявках на проверку объектов гражданской инфраструктуры:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор проверки (целое число, автоинкремент); |
| * «address» - | Адрес, который требуется проверить (строка длиной до 255 символов, обязательное поле); |
| * «status\_id» - | Статус проверки (целое число), внешний ключ, ссылающийся на поле id в таблице Statuses; |
| * «inspector\_id» - | Сотрудник ответственный за проверку (целое число), внешний ключ, ссылающийся на поле id в таблице Employees; |
| * «plan\_date» - | Плановая дата проверки (тип DATE, обязательное поле). |

Листинг 9 – Создание таблицы таблицы с бригадами

CREATE TABLE Brigades (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL,

phone\_number VARCHAR(15) NOT NULL

);

В таблице Brigades описаны поля с информацией о бригадах:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор пожарной бригады (целое число, автоинкремент); |
| * «name» - | Название пожарной бригады (строка длиной до 100 символов, обязательное поле); |
| * «Address» - | Адрес пожарной бригады (строка длиной до 255 символов, обязательное поле); |
| * «phone\_number» - | Номер телефона пожарной бригады (строка длиной до 15 символов, обязательное поле). |

Листинг 10 – Создание таблицы с вызовами

CREATE TABLE Call (

id INT PRIMARY KEY AUTO\_INCREMENT,

address VARCHAR(255) NOT NULL,

user\_id INT,

date\_call DATETIME NOT NULL,

status\_id INT,

comment TEXT,

fire\_source VARCHAR(255),

hazard\_id INT,

complexity\_id INT,

FOREIGN KEY (user\_id) REFERENCES Users(id),

FOREIGN KEY (status\_id) REFERENCES Statuses(id),

FOREIGN KEY (hazard\_id) REFERENCES Hazards(id),

FOREIGN KEY (complexity\_id) REFERENCES Complexity(id)

);

В таблице Call описаны поля для хранения информации о вызовах экстренных служб:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор вызова пожарной службы (целое число, автоинкремент); |
| * «address» - | Место возгорания (строка длиной до 255 символов, обязательное поле); |
| * «user\_id» - | Уникальный идентификатор пользователя, который совершил вызов пожарной службы (целое число), внешний ключ, ссылающийся на поле id в таблице Users; |
| * «date\_call» - | Дата вызова (тип DATETIME, обязательное поле); |
| * «status\_id» - | Статус, в котором находится вызов (целое число), внешний ключ, ссылающийся на поле id в таблице Statuses; |
| * «comment» - | Комментарий пользователя (тип TEXT, необязательное поле); |
| * «fire\_source» - | Источник пожара (строка длиной до 255 символов, необязательное поле); |
| * «hazard\_id» - | Уникальный идентификатор опасного фактора (целое число), внешний ключ, ссылающийся на поле id в таблице Hazards; |
| * «complexity\_id» - | Сложность пожара (целое число), внешний ключ, ссылающийся на поле id в таблице Complexity. |

Листинг 11 – Создание связующей таблицы для хранения информации о принадлежности сотрудников к бригадам

CREATE TABLE Employees\_brigades (

id INT PRIMARY KEY AUTO\_INCREMENT, brigade\_id INT NOT NULL,

employee\_id INT NOT NULL,

FOREIGN KEY (brigade\_id) REFERENCES Brigades(id),

FOREIGN KEY (employee\_id) REFERENCES Employees(id)

);

В таблице Employees\_brigades описаны поля для связи сотрудников и бригад:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор записи (целое число, автоинкремент); |
| * «brigade\_id» - | Уникальный идентификатор бригады (целое число, обязательное поле), внешний ключ, ссылающийся на поле id в таблице Brigades; |
| * «employee\_id» - | Уникальный идентификатор сотрудника (целое число, обязательное поле), внешний ключ, ссылающийся на поле id в таблице Employees. |

Листинг 12 – Создание связующей таблицы для хранения информации о бригадах, выезжавших на вызов

CREATE TABLE Brigades\_call (

id INT PRIMARY KEY AUTO\_INCREMENT,

brigade\_id INT NOT NULL,

call\_id INT NOT NULL,

FOREIGN KEY (brigade\_id) REFERENCES Brigades(id),

FOREIGN KEY (call\_id) REFERENCES Call(id)

);

В таблице Brigades\_call описсаны поля для связи бригад задействованных на вызовах:

|  |  |
| --- | --- |
| * «id» - | Уникальный идентификатор записи (целое число, автоинкремент); |
| * «brigade\_id» - | Уникальный идентификатор бригады (целое число, обязательное поле), внешний ключ, ссылающийся на поле id в таблице Brigades; |
| * «call\_id» - | Уникальный идентификатор вызова (целое число, обязательное поле), внешний ключ, ссылающийся на поле id в таблице Call. |

Таким образом база данных для системы автоматического оповещения МЧС о чрезвычайных ситуациях создана

## Разработка мобильного приложения

### Разработка View элементов

View - элементы разработаны на двух языках:

* C# - для обработки событий нажатия на экран пользователем;
* XAML - для верстки оконных форм.

Для авторизации пользователя разработана оконная форма с логотипом, поле ввода логина и пароля (Entry), кнопка «Войти» (Button), текстовые элементы (Label) «Зарегистрироваться» и «Вызвать звонком» с обработкой события нажатия (Листинг 13).

Листинг 13 – Отображение элементов окна авторизации

<StackLayout

WidthRequest="300"

HorizontalOptions="Center"

VerticalOptions="CenterAndExpand">

<Image

Source="DiplomPRoject\Resources\Images\logo.png" HeightRequest="75"

WidthRequest="75"

Margin="0,20,0,50"/>

<Entry Placeholder="Введите логин"

Keyboard="Email"

HorizontalOptions="Center"

TextColor="Black"/>

<Entry Placeholder="Введите пароль"

IsPassword="True"

HorizontalOptions="Center"

TextColor="Black"/>

<Button Text="Войти"

Clicked="LoginButton\_Clicked"

BackgroundColor = "Green"

TextColor="White"/>

<Label Text="Зарегестрироваться"

TextDecorations="Underline"

TextColor="Blue"

HorizontalOptions="Center"

VerticalOptions="CenterAndExpand"

FontSize="16"

Margin="0,0,0,0"/>

<Label Text="Вызвать звонком"

TextDecorations="Underline"

TextColor="Blue"

HorizontalOptions="Center"

VerticalOptions="CenterAndExpand"

FontSize="16"

Margin="0,20,0,0"/>

</StackLayout>

Для выбора местоположения пользователя разработана оконная форма с отображением Яндекс - карт. Для отображения карты добавлен WebView - элемент, настройка которого выполнена в файле Main.xaml.cs (Листинг 14).

Листинг 14 – WebView для отображения Яндекс карт

private void LoadMap()

{

// Загрузка HTML-кода с картой Яндекса

string html = @"

<!DOCTYPE html>

<html>

<head>

<meta name='viewport' content='initial-scale=1.0, user-scalable=no'>

<meta charset='utf-8'>

<style>

/\* Установите размеры контейнера карты \*/

#map {

height: 100%;

}

html, body {

height: 100%;

margin: 0;

padding: 0;

}

</style>

<script src='https://api-maps.yandex.ru/2.1/?apikey=a6da2898-c703-4e40-a1bd-440682600465'></script>

<script>

ymaps.ready(init);

function init(){

var myMap = new ymaps.Map('map', {

center: [55.753215, 37.622504], // Координаты центра карты

zoom: 9 // Уровень масштабирования

});

}

</script>

</head>

<body>

<div id='map'></div>

</body>

</html>";

webView.Source = new HtmlWebViewSource { Html = html };

}

### Разработка Presenter для хранения и обработки данных

Элементы Presenter разработаны на объектно - ориентированном C# для хранения и обработки данных, отображающихся на элементе View.

Для окна авторизации разработан класс SignInPresenter (Листинг 15). Методы класса обрабатывают введенные данные логина и пароля пользователя, отправляют их на сервер и открывают следующий View – элемент. При ошибке на сервере метод HandleServerError вызывает метод класса ShowErrorMessage (View элемент окна авторизации), который отображает всплывающее окно, с ошибкой авторизации. Метод OpenPhoneApp() открывает встроенное приложение «Телефон» с автоматически введенным номером «112», в случае отсутствия интернет - соединения.

Листинг 15 - Presenter окна авторизации

public class SignInPresenter : ISignInPresenter

{

private ISignInView \_view;

private ISignInInteractor \_interactor;

private string \_username;

private string \_password;

public SignInPresenter()

{

}

public SignInPresenter(string username)

{

\_username = username;

}

public void SetInteractor(ISignInInteractor interactor)

{

\_interactor = interactor;

}

public void SetView(ISignInView view)

{

\_view = view;

}

public void SetUsername(string username)

{

\_username = username;

}

public void SetPassword(string password)

{

\_password = password;

}

public async void OnSignInButtonClicked()

{

if (string.IsNullOrWhiteSpace(\_username) || string.IsNullOrWhiteSpace(\_password))

{

\_view.ShowErrorMessage("Username and password cannot be empty.");

return;

}

var (isValid, errorMessage) = await \_interactor.SignIn (\_username, \_password);

if (isValid)

{

HandleSuccessfulLogin();

}else{

HandleServerError(errorMessage);

}

}

public void HandleServerError(string errorMessage)

{

\_view.ShowErrorMessage(errorMessage);

}

public void HandleSuccessfulLogin()

{

\_view.ShowSuccessMessage();

}

public void OpenPhoneApp()

{

OpenPhoneApp();

}

public void NavigateToRegistration()

{

NavigateToRegistration();

}

}

### Разработка Interactor для организации запросов к серверу

Элемент интерактор описан в APIClient (Листинг 16), отвечает за организацию запросов к серверу и получению ответов от него.

Листинг 16 – Файл APIClient

using System;

using System.Net.Http;

using System.Threading.Tasks;

using Newtonsoft.Json;

public interface IAPIClient

{

Task<DataRequest> RequestAsync<Request>(Request request, Action<Request.Response, Exception, HttpStatusCode?> completion)

where Request : IEndPointType;

}

public class APIClient : IAPIClient

{

private readonly HttpClient \_httpClient;

public APIClient()

{

\_httpClient = new HttpClient();

}

public async Task<DataRequest> RequestAsync<Request>(Request request, Action<Request.Response, Exception, HttpStatusCode?> completion)

where Request : IEndPointType

{

try

{

var httpRequest = new HttpRequestMessage(request.Method, request.Url);

if (request.HttpBody != null)

{

var jsonBody = JsonConvert.SerializeObject(request.HttpBody);

httpRequest.Content = new StringContent(jsonBody, System.Text.Encoding.UTF8, "application/json");

}

var response = await \_httpClient.SendAsync(httpRequest);

var httpStatusCode = response.StatusCode;

if (response.IsSuccessStatusCode)

{

var responseBody = await response.Content.ReadAsStringAsync();

var responseObject = JsonConvert.DeserializeObject<Request.Response>(responseBody);

completion(responseObject, null, httpStatusCode);

}

else

{

completion(default(Request.Response), new Exception($"HTTP Error: {httpStatusCode}"), httpStatusCode);

}

}

catch (Exception ex)

{

completion(default(Request.Response), ex, null);

}

return null;

}

}

public interface IEndPointType

{

string Url { get; }

HttpMethod Method { get; }

object HttpBody { get; }

}

public class EndPointType<TResponse> : IEndPointType

{

public string Url { get; }

public HttpMethod Method { get; }

public object HttpBody { get; }

public EndPointType(string url, HttpMethod method, object httpBody)

{

Url = url;

Method = method;

HttpBody = httpBody;

}

}

### Разработка Entity для хранения данных

Для хранения информации, получаемой из базы данных, созданы структуры:

* 1. Для хранения информации об авторизованном пользователе разработана структура UserInfo. Автосвойства структуры: FirstName (Имя пользователя), LastName (Фамилия пользователя), MiddleName (Отчество), HomeAddress (Домашний адрес), PhoneNumber (Номер телефона). Также для отладки кода добавлена перегрузка метода ToString (Листинг 17).

Листинг 17 - Структура UserInfo

public struct UserInfo

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string MiddleName { get; set; }

public string HomeAddress { get; set; }

public string PhoneNumber { get; set; }

public UserInfo(string firstName, string lastName, string middleName, string homeAddress, string phoneNumber)

{

FirstName = firstName;

LastName = lastName;

MiddleName = middleName;

HomeAddress = homeAddress;

PhoneNumber = phoneNumber;

}

public override string ToString()

{

return $"Имя: {FirstName}, Фамилия: {LastName}, Отчество: {MiddleName}, Адрес: {HomeAddress}, Телефон: {PhoneNumber}";

}

}

1. Для хранения информации о вызове экстренных служб разработана структура CallInfo. Автосвойства структуры: Adress (Адрес), UserId (Id пользователя, совершившего вызов), CallDate (Дата вызова), CallStatus (Домашний адрес), Comment (Номер телефона). Также для отладки кода добавлена перегрузка метода ToString (Листинг 18).

Листинг 18 - Структура Call

public struct CallInfo

{

public string Address { get; set; }

public int UserId { get; set; }

public DateTime CallDate { get; set; }

public string CallStatus { get; set; }

public string Comment { get; set; }

public CallInfo(string address, int userId, string callDate, string callStatus, string comment)

{

Address = address;

UserId = userId;

if (!DateTime.TryParse(callDateString, out DateTime callDate))

{

throw new ArgumentException("Invalid date format", nameof(callDateString));

}

CallDate = callDate;

CallStatus = callStatus;

Comment = comment;

}

public override string ToString()

{

return $"Адрес: {Address}, ID Пользователя: {UserId}, Дата Вызова: {CallDate}, Статус Вызова: {CallStatus}, Комментарий: {Comment}";

}

}

## Вывод по разделу

В разделе описана разработка базы данных на PostgreSQL и кроссплатформенного мобильного приложения для системы автоматического оповещения экстренных служб.

# Социально-экономическая эффективность

Система автоматизированного вызова экстренных служб с помощью мобильного приложения является социально-экономическим проектом ускорения принятия решений в эффективности оказания экстренной помощи в чрезвычайных ситуациях.

## Социальная эффективность

### Ускорение реагирования на чрезвычайные ситуации

Автоматическое определение местоположения пользователя и отправка точных координат значительно ускоряет процесс вызова помощи, особенно в ситуациях, когда человек не может точно описать свое местоположение.

Использование технологии GPS исключает ошибки, связанные с человеческим фактором, при передаче координат и описания ситуации

### Повышение безопасности населения

Быстрая и точная передача информации о происшествии, наряду с наличием удобного и надежного инструмента для вызова помощи, значительно сокращает время прибытия экстренных служб, что особенно важно в критических ситуациях и способствует улучшению общего уровня безопасности и уверенности населения.

### Инклюзивность

В проект заложены возможности адаптации для использования людьми с различными видами инвалидности, что повышает их возможности в экстренных ситуациях.

Возможность использования приложения на разных языках обеспечивает доступность для иностранцев и людей, не владеющих национальным языком.

## Экономическая эффективность

### Снижение затрат на экстренные службы

Точные координаты и предварительная информация о ситуации позволяют экстренным службам более эффективно планировать свои действия, что снижает затраты на топливо и амортизацию транспортных средств.

### Косвенные экономические выгоды

Быстрая и эффективная помощь в экстренных ситуациях может предотвратить серьезные последствия (например, инвалидность, смертность), что в свою очередь снижает экономические потери для общества и системы здравоохранения.

## Определение капитальных вложений

К капитальным вложениям на разработку программного обеспечения автоматизированной системы вызова экстренных служб относятся затраты на приобретение технических средств, программных средств, транспортировку и разработку.

Расчет капитальных вложений производим по формуле:

, (1)

Где ЦТС – цена приобретения технических средств, руб.;

ЦПС – цена приобретения программных средств, руб.;

Н – накладные расходы, руб.;

СР – стоимость разработки программного обеспечения, руб.

Цена приобретения технических средств равна нулю, так как дополнительная закупка не требуется, поскольку разработка велась на уже имеющихся средствах компании.

Цена приобретения программных средств, составляется из покупки лицензии на использования Яндекс API, для использования карт, определяющих местоположения пользователя. Цена использования Яндекс API равна 150000 руб. в год для 1000 запросов в год, что достаточно для разработки.

Накладные расходы равны нулю, поскольку не требуется дополнительных приобретений.

Стоимость разработки автоматизированной системы вызова экстренных служб формируется из затрат на оплату труда работников, равную 90000 руб./мес. На разработку потребовалось 3 специалиста работавших в течении двух месяцев, следовательно, стоимость разработки составляет 270000 руб.

Расчет капитальных затрат составляет:

руб.

## Расчет экономического эффекта

Внедрение автоматизированной системы оповещения экстренных служб о чрезвычайных ситуациях, увеличивает скорость обработки одной заявки, в среднем, на 30 секунд (с двух минут до полутора минут), следовательно, рабочее время, затраченное оператором, уменьшается на 25%. Благодаря этому, уменьшаются затраты на обработку одной заявки, а также уменьшается затраты в целом на заработную плату каждого оператора на 25%.

Пусть каждый месяц совершается V – количество вызовов. Заработная плата одного оператора равна P - рублей/месяц. При количестве операторов n – человек, затраты на их заработную плату составляют:

|  |  |  |
| --- | --- | --- |
|  | рублей/месяц | (1) |

Стоимость одного вызова составляет:

|  |  |  |
| --- | --- | --- |
|  | рублей | (2) |

При использовании системы автоматизированного оповещения МЧС о чрезвычайных ситуациях стоимость одного вызова составит:

|  |  |  |
| --- | --- | --- |
|  | рублей | (3) |

Таким образом стоимость вызова экстренных служб с использованием системы автоматизированного оповещения МЧС в K-раз меньше, чем текущий способ вызова:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

По статистике за 2023 год в Пермском Крае среднее количество вызовов пожарной службы в месяц равно 249, а всего сотрудников, обрабатывающих вызовы, равно 80 человек с ежемесячной заработной платой равной 35000 рублей.

Таким образом, по формуле (4) рассчитано во сколько раз использование автоматизированной системы сокращает затраты на заработную плату операторов службы МЧС:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Ежемесячные затраты на оплату труда оператор Пермского Края рассчитаны по формуле (6) без использования автоматизированной системы оповещения экстренных служб и по формуле (7) - с использованием:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |
|  |  | (7) |

Таким образом система автоматизированного оповещения экстренных служб окупается в течении первого месяца, и далее обеспечивает ежемесячную экономию средств на заработную плату операторов в размере 646 153 руб.

Таким образом, пожарные службы МЧС по Пермскому Краю получат прибыль в виде экономии денежных средств на заработную плату операторов:

* в первый месяц – 226 153 рублей;
* в остальные месяцы года – 7 107 683 рублей;
* в течение первого года эксплуатации системы – 7 333 836 рублей;
* в течении следующих лет по – 7 753 836 рублей/год.

## Вывод по разделу

Внедрение мобильного приложения для вызова экстренных служб с имеет значительный социально-экономический эффект, способствует повышению безопасности и качества жизни населения, снижению затрат на работу экстренных служб и предотвращению экономических потерь от несчастных случаев.

Рассчитана стоимость разработки автоматизированной системы оповещения экстренных служб о чрезвычайных ситуациях, равная 420000. Срок окупаемости разработки составляет один месяц.

Таким образом, в первом году использования системы экономия денежных средств на заработную плату операторов составляет 7 333 836 рублей, а в последующие годы - по 7 753 836 рублей.

# Информационная безопасность

В системе автоматизированного вызова экстренных служб, реализовано хранение паролей с использованием метода хеширования с «солью», с применением алгоритма SHA-256.

Процесс хэширования паролей с солью состоит из нескольких этапов, которые обеспечивают дополнительный уровень защиты. Основные шаги включают:

1. Генерация соли: При создании или изменении пароля для каждого пользователя генерируется уникальная соль. Соль представляет собой случайную строку данных, которая добавляется к паролю перед хэшированием. Это предотвращает использование заранее вычисленных таблиц хэшей (таблицы радужных цепей).
2. Соединение пароля и соли: Соль добавляется к паролю пользователя. Например, если пароль пользователя – "password123", а сгенерированная соль – "s@1tValu3", то результирующая строка перед хэшированием будет "password123s@1tValu3".
3. Хэширование строки: Объединенная строка пароля и соли подвергается хэшированию с использованием алгоритма SHA-256. Алгоритм SHA-256 генерирует хэш длиной 256 бит (32 байта).
4. Сохранение хэша и соли: Хэш и соль сохраняются в базе данных. При этом исходный пароль нигде не хранится. Хранятся только хэш и соответствующая соль, что делает восстановление исходного пароля невозможным без знания соли и исходного пароля.

При аутентификации пользователя его введённый пароль проходит через тот же процесс хэширования с использованием «соли», сохраненной в базе данных. Результирующий хэш сравнивается с хранимым хэшем пароля. Если хэши совпадают, аутентификация считается успешной.

## Вывод по разделу

Использование метода хэширования паролей с «солью» и алгоритма SHA-256 обеспечивает надежную защиту пользовательских паролей в системе. Это предотвращает ряд атак, включая атаки методом подбора и атаки с использованием предварительно вычисленных таблиц хэшей. Данная методология строго соблюдена для обеспечения безопасности пользовательских данных и целостности системы.

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы поставленная цель достигнута.

Для достижения цели решены следующие задачи:

* проанализированы существующие системы вызова экстренных служб;
* спроектирована архитектура пользовательского взаимодействия, разделение по ролям и построены связи между ролями;
* проанализированы метода структуризации кода;
* выбран Framework для разработки;
* разработан интерфейс мобильного приложения;
* разработано мобильное приложение для вызова экстренных служб.

Проведен расчет стоимости и времени окупаемости проекта.

В разделе по информационной безопасности решена задача защищенного хранения паролей пользователей в базе данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация .net MAUI [Электронный ресурс] URL: Основные возможности https://learn.microsoft.com/ru-ru/dotnet/maui/what-is-maui?view=net-maui-8.0 (Дата обращения 26.09.2034).
2. Руководство по C# [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (Дата обращения 14.12.2023).
3. Локальные нормативные акты Главного управления МЧС России по Пермскому Краю [Электронный ресурс] URL: https://59.mchs.gov.ru/documents/lokalnye-normativnye-akty-glavnogo-upravleniya-mchs-rossii-po-permskomu-krayu (Дата обращения 09.10.2023).
4. Локальные нормативные акты Главного управления МЧС России по Пермскому Краю [Электронный ресурс] URL: https://59.mchs.gov.ru/documents/lokalnye-normativnye-akty-glavnogo-upravleniya-mchs-rossii-po-permskomu-krayu (Дата обращения 09.10.2023).
5. Федеральные конституционные законы [Электронный ресурс] URL: https://mchs.gov.ru/dokumenty/federalnye-konstitucionnye-zakony
6. Своды правил [Электронный ресурс] URL: https://59.mchs.gov.ru/documents/svody-pravil (Дата обращения 09.10.2023).
7. Документация для разработчиков — API Яндекс Карты [Электронный ресурс] URL: https://yandex.ru/dev/jsapi30/doc/ru/ (Дата обращения 25.03.2024).
8. Создание API-интерфейсов RESTful [Электронный ресурс]  URL: https://learn.microsoft.com/ru-ru/aspnet/web-api/overview/older-versions/build-restful-apis-with-aspnet-web-api (Дата обращения 13.01.2024).