МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к дипломному проекту:**

Система контроля и анализа эвакуации в условиях чрезвычайных ситуаций

Дипломник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тышкевич Р.А.

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жиляк Н.А., к.т.н.,доцент

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Блинова Е. А., к.т.н., доцент

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соболевский А. С., ст. преп.

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Николайчук А.Н., пред.-стажер

Дипломный проект защищен с оценкой

Председатель ГЭК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2025

**Содержание**

Введение 3

1. Постановка задачи и анализ аналогичных решений 3

1.1. Постановка задачи 3

1.2. Актуальность задачи 3

1.3. Обзор аналогов 3

1.3.1. Аналог 1 3

1.3.2. Аналог 2 3

1.3.3. Аналог 3 3

1.4. Анализ решений 3

1.4.1. Требование к приложению 3

1.4.2. Выбор библиотек и технологий для серверной стороны 3

1.4.3. Выбор библиотек и технологий для клиентской стороны 3

1.4.4. Выбор средств программирования 3

1.5. Выводы по разделу 3

2. Проектирование приложения 3

2.1. Диаграмма вариантов использования 3

2.1.1. Алгоритм 1 3

2.2. Архитектура приложения 3

2.3. Проектирование базы данных 3

2.4. Выводы по разделу 4

3. Реализация веб-приложения 4

3.1. Разработка серверной части приложения 4

3.2. Подключение базы данных 4

3.3. Реализация клиентской части приложения 4

3.4. Выводы по разделу 4

4. Анализ информационной безопасности приложения 4

4.1. Выводы по разделу 4

5. Тестирование приложения 4

6. Руководство программиста 4

6.1. Роль студента 4

6.2. Роль преподавателя 4

6.3. Выводы по разделу 4

7. Технико-экономическое обоснование проекта 4

7.1. Общая характеристика разрабатываемого программного средства 4

7.2. Исходные данные и маркетинговый анализ 4

7.3. Методика обоснования цены 4

7.3.1. Определение объёма программного средства 4

7.3.2. Основная заработная плата 4

7.3.3. Дополнительная заработная плата 5

7.3.4. Расчёт отчислений на социальные цели 5

7.3.5. Расходы на материалы 5

7.3.6. Расходы на специальное оборудование и платные услуги 5

7.3.7. Расчет прочих прямых затрат 5

7.3.8. Расчет общепроизводственных и общехозяйственных расходов 5

7.3.9. Расчет суммы расходов на разработку программного средства 5

7.3.10. Расходы на сопровождение и адаптацию 5

7.3.11. Расчет общей суммы расходов 5

7.3.12. Определение цены, оценка эффективности 5

7.4. Выводы по разделу 5

# Введение

Наступил век технологий, когда эффективность принятия решений и оперативность реагирования в чрезвычайных ситуациях напрямую зависят от надёжности и функциональности цифровых инструментов. В современных условиях именно интегрированные веб-сервисы и боты в мессенджерах становятся ключевыми компонентами систем, обеспечивающих координацию эвакуации, информирование населения и сбор аналитических данных в реальном времени.

Дипломный проект представляет собой комплексное решение — веб-приложение и Telegram-бот, призванные автоматизировать процессы контроля и анализа эвакуации в условиях чрезвычайных ситуаций. Основная цель проекта – предоставить экстренным службам и ответственным лицам интуитивно понятный и надёжный инструмент для:

* формирования оперативных планов эвакуации;
* рассылки уведомлений и инструкций населению;
* приёма и обработки запросов на эвакуацию от граждан;
* сбора статистики и последующего анализа эффективности мероприятий.

Интерфейс веб-приложения разработан на платформе ASP .NET и обеспечивает:

* авторизацию пользователей с разными уровнями доступа;
* визуализацию оперативной карты с текущим статусом районов и маршрутов эвакуации;
* модуль управления списками эвакуируемых и распределения ресурсов;
* панель аналитики для построения отчётов по выполненным операциям.

Для взаимодействия через мессенджер реализован Telegram-бот на базе . NET, который поддерживает:

* отправку инструкций и предупреждений в зависимости от зоны риска;
* оперативное уведомление ответственных служб о новых инцидентах.

В качестве СУБД используется PostgreSQL, обеспечивающая надёжное хранение пространственных и табличных данных. Для упрощения развёртывания и масштабирования проекта весь стек компонентов (веб-приложение, бот, база данных) упакован в Docker-контейнеры.

Цели проекта:

* Исследовать существующие решения в области цифровых систем эвакуации и проанализировать их сильные и слабые стороны.
* Спроектировать архитектуру серверной и клиентской частей приложения с учётом надёжности и отказоустойчивости.

Провести функциональное и нагрузочное тестирование готового решения.

# Постановка задачи и анализ аналогичных решений

# Постановка задачи

Цель проекта — разработка программной системы, предназначенной для контроля, анализа и информационного сопровождения эвакуационных процессов в условиях чрезвычайных ситуаций. Система должна обеспечивать визуализацию опасных зон на интерактивной карте, предоставление пользователям актуальной информации о маршрутах эвакуации, а также оперативное оповещение через Telegram-бота с рекомендациями по действиям в экстренных ситуациях.

Для достижения поставленной цели необходимо решить следующие задачи:

* Разработать веб-приложение, включающее:
  + Интерактивную карту Республики Беларусь с возможностью отображения опасных зон в реальном времени;
  + Систему визуальной маркировки территорий с различным уровнем риска (например, цветовое выделение зон);
  + Механизм динамического построения безопасных маршрутов с учётом текущей обстановки;
  + Удобный и понятный пользовательский интерфейс, ориентированный на широкий круг пользователей.
* Разработать Telegram-бота, обеспечивающего:
  + Оповещение пользователей о наступлении чрезвычайных ситуаций;
  + Автоматическую рассылку инструкций, рекомендаций и схем эвакуации;
  + Персонализированную информацию в зависимости от местоположения или привязанного объекта пользователя (например, университет, жилой район и др.);
  + Возможность интеграции с веб-приложением для получения актуальных данных об опасных зонах.
* Обеспечить архитектуру системы, которая:
  + Позволяет легко масштабировать проект и дополнять его новыми функциями;
  + Поддерживает централизованное и своевременное обновление информации;
  + Гарантирует надёжность и отказоустойчивость при передаче данных между компонентами системы.

Реализация проекта направлена на повышение уровня готовности и информированности населения в условиях чрезвычайных ситуаций, а также на поддержку эффективной и безопасной эвакуации с помощью современных информационных технологий.

# Актуальность задачи

фывфыв

# Обзор аналогов

На текущий момент существует множество различных систем для организации и контроля эвакуации в условиях чрезвычайных ситуаций. Для формирования требований к функционалу разрабатываемой системы необходимо рассмотреть достоинства и недостатки аналогов. В качестве аналогов разрабатываемой системы можно выделить список следующих:

* система МЧС России;
* система МЧС Беларуси;
* система МЧС Казахстана.

# Официальный сайт МЧС России

Сайт МЧС России – это удобное решение, предназначенное для контроля и анализа эвакуации в условиях чрезвычайных ситуаций. На главной странице сайта доступна фильтрация данных по типу чрезвычайной ситуации и дате, а также функции авторизации и регистрации пользователей, редактирования профиля и просмотра текущих, завершенных и отмененных операций по эвакуации. Главная страница системы представлена на рисунке 1.1. Также в системе реализована возможность авторизации пользователя. Главная страница находится на рисунке 1.1.

Изображение выглядит как текст, снимок экрана, Веб-сайт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.1 – Главная страница сайта МЧС России

Также на сайте представлена возможность просмотра оперативной карты. Оперативная карта представлена на рисунке 1.2.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.2– Раздел «Оперативной информации»

В интерфейсе системы используются только три цвета, которые видит пользователь. На рисунке 1.1 можно наблюдать, что при разработке интерфейса применяются красный, белый и синий цвета, однако следует отметить, что дизайн системы ориентирован на удобство использования с мобильных устройств, что подтверждают характерные закругленные по бокам кнопки. Также стоит отметить, что на одной и той же странице текст отображается разными цветами.

К достоинствам системы можно отнести:

* отсутствие рекламы;
* наличие личного аккаунта;
* возможность редактирования профиля;
* уведомления об изменениях в системе.

К недостаткам системы можно отнести следующие:

* длительное время обработки запросов на обновление данных, равное 3–5 секундам;

отображение завершенных операций на главной странице при поиске и анализе текущих данных

# Официальный сайт МЧС Беларуси

Данный веб-сайт МЧС Республики Беларусь представляет собой типовой государственный портал, стилистически напоминающий сайты 90-х годов. На сайте реализована авторизация пользователей. Также предусмотрена функция быстрого поиска расписания приёма граждан.

Главная страница веб-сайта МЧС Беларуси представлена на рисунке 1.3. На рисунке видно не совсем удобный дизайн: простой фон, избыточные навигационные блоки и несогласованные между собой элементы, что затрудняет восприятие и фокусировку на ключевых разделах.

Из этого следует, что визуальные компоненты играют критически важную роль для комфортного взаимодействия с системой экстренного ведомства.

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.3– Главная страница веб-сайта МЧС Беларуси

На странице, есть возможность просмотра графика приема граждан нажав на соответствующую кнопку в меню выше.

При нажатии на нее мы переходим на страницу, которая имеет весьма схожий дизайн, выполненный в стиле главной страницы, график представлен на рисунке 1.4.

Изображение выглядит как текст, снимок экрана, веб-страница, Веб-сайт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.4– Страница веб-сайта МЧС для просмотра графика приема граждан

На главной странице доступна кнопка перехода на оперативную карту – при нажатии пользователь попадает на страницу с интерактивной картой Республики Беларусь (рисунок 1.5).

Изображение выглядит как текст, снимок экрана, карта

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.5– Оперативная карта.

**К достоинствам сайта можно отнести:**

* отсутствие рекламных блоков;
* функционал управления записями на приём;
* адаптивность страницы расписания для разных устройств;
* уведомления пользователей об изменениях в графике приёма.

**К недостаткам сайта можно отнести:**

* устаревший визуальный стиль главной страницы;
* отсутствие возможности редактирования профиля;
* избыточное количество отвлекающих элементов вне раздела расписания

# Официальный сайт МЧС Казахстана

Данный веб-сайт МЧС Республики Казахстан выполнен в лаконичном и ненагруженном стиле: интуитивно понятный интерфейс и быстрая навигация позволяют оперативно записаться на приём.

Главная страница приложение представлена на рисунке 1.6.

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки. Рисунок 1.6– Главная страница сайта МЧС Казахстана

Обращает на себя внимание грамотно подобранная цветовая гамма сайта, цвета приятные для глаза, а так же не сильно нагружают внешний вид сайта.

При просмотре графика приема (рисунок 1.7) можно заметить, что соблюдается общая цветовая гамма, расположение лиц осуществляющих прием граждан выполнена удобно, имеются фотографии, а также указаны номера телефонов.

Изображение выглядит как текст, программное обеспечение, веб-страница, Веб-сайт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.7– Страница приема граждан.

**К достоинствам сайта можно отнести:**

* отсутствие рекламных блоков;
* возможность управления записями через личный кабинет;
* приятный и минималистичный дизайн страниц.

**К недостаткам можно отнести:**

* отсутствие функции редактирования личного профиля;
* отсутствие уведомлений пользователю об изменениях в графике приёма.

# Анализ решений

При создании программного средства одним из важных моментов является выбор языка программирования, так как от него зависит множество факторов, таких как скорость создания программы, скорость и удобство тестирования, возможность миграции на другие платформы, возможность быстрого внесения изменений и так далее. Также стоит понимать, что идеального языка программирования нет, так как все разные инструменты и имеют свои преимущества и недостатки при разработке. Наиболее актуальными сегодня являются Java, C#, Python, JavaScript.

# Требование к приложению

Разработчиком при создании программной системы контроля и анализа эвакуации в условиях чрезвычайных ситуаций были установлены следующие требования к функциональности компонентов:

Веб-приложение должно обеспечивать**:**

* отображение интерактивной карты территории Республики Беларусь;
* визуальное выделение опасных зон (например, при помощи цветовой маркировки: жёлтая — нежелательное пребывание, красная — закрытая зона);
* отображение текущей информации о чрезвычайных ситуациях в режиме реального времени;
* построение безопасных маршрутов с учётом опасных зон;
* отображение рекомендаций и инструкций по эвакуации в зависимости от местоположения;
* возможность обновления и расширения данных об опасных зонах через административный интерфейс.

Telegram-бот должен обеспечивать:

* регистрацию и авторизацию пользователя в системе;
* рассылку уведомлений при возникновении чрезвычайных ситуаций;
* отправку текстовых и графических инструкций по эвакуации (в том числе схем эвакуации зданий);
* персонализированные уведомления в зависимости от привязанного к пользователю объекта (например, университет, школа и т.д.);
* возможность получения справочной информации по действиям в различных ЧС.

Общие требования к системе:

* устойчивость к нагрузке при массовой отправке уведомлений;
* синхронизация данных между веб-приложением и Telegram-ботом;
* возможность масштабирования системы (добавление новых регионов, объектов, типов ЧС);
* обеспечение безопасности пользовательских данных;

удобный и понятный интерфейс для пользователей и администраторов.

# Выбор библиотек и технологий для серверной стороны

Для реализации серверной части системы контроля и анализа эвакуации были выбраны современные и надёжные технологии, обеспечивающие стабильную работу, масштабируемость и удобство в разработке.

В качестве языка программирования используется Java версии 23, что позволяет применять новейшие языковые конструкции и гарантирует долгосрочную поддержку. Для управления сборкой и зависимостями выбран инструмент Gradle с использованием Groovy DSL, что обеспечивает гибкость конфигурации и простоту автоматизации.

Серверная часть выполняет функции API-сервиса и взаимодействует как с веб-приложением, так и с Telegram-ботом. Для разработки REST API применяется фреймворк Spring Boot, благодаря которому упрощается процесс настройки и ускоряется разработка, при этом обеспечивается высокая производительность приложения.

Дополнительно используются следующие технологии и библиотеки:

* Spring Web — для обработки HTTP-запросов и маршрутизации;
* Spring Security — для обеспечения авторизации и аутентификации;
* Spring Data JPA + Hibernate — для работы с базой данных;
* PostgreSQL — в качестве реляционной СУБД, хранящей информацию о пользователях, зонах риска, уведомлениях и т.д.;

# Выбор библиотек и технологий для клиентской стороны

Клиентская сторона представлена в виде веб-приложения, основной задачей которого является отображение интерактивной карты с актуальной информацией об опасных зонах, маршрутах и рекомендациях по эвакуации.

Для реализации веб-интерфейса выбраны следующие технологии:

* HTML5 и CSS3 — для создания структуры и стилизации страниц;
* JavaScript — для реализации клиентской логики;
* React — как основной JavaScript-фреймворк для построения интерфейса с возможностью реактивного обновления компонентов;
* Leaflet.js — библиотека для работы с интерактивными картами, позволяющая отображать географические объекты и выделять зоны риска;
* Axios — для выполнения асинхронных HTTP-запросов к серверному API;
* Bootstrap (либо Tailwind CSS) — для ускоренной разработки адаптивного интерфейса.

Веб-клиент обеспечивает пользователю быстрый доступ к актуальной информации, возможность взаимодействия с картой и получение маршрутов в удобном и интуитивно понятном виде.

# Выбор средств программирования

В процессе реализации проекта используются современные средства программирования, обеспечивающие удобную разработку, отладку и сопровождение системы.

Основные средства разработки:

* IntelliJ IDEA — интегрированная среда разработки для Java, предоставляющая мощные инструменты для работы со Spring Boot, Gradle и другими фреймворками;
* Visual Studio Code — лёгкий и удобный редактор для разработки клиентской части с поддержкой React и JavaScript;
* PostgreSQL + pgAdmin — для управления базой данных;
* Git + GitHub — для контроля версий и совместной работы;
* Docker (опционально) — для контейнеризации и развертывания компонентов системы в разных средах;
* ngrok — для локального тестирования Telegram-бота с внешним доступом.

Выбор указанных средств обусловлен их популярностью, открытым исходным кодом, активным сообществом и богатым функционалом, что значительно упрощает процесс реализации дипломного проекта.

# Выводы по разделу

В данном разделе был проведен анализ предметной области по теме «Система контроля и анализа эвакуации в условиях чрезвычайных ситуаций». Также были рассмотрены некоторые аналоги государственных систем по эвакуации и информированию населения в случае ЧС.

При написании проекта были учтены как положительные, так и отрицательные черты аналогов. Был изучен опыт других разработчиков. Проанализировав сильные и слабые стороны приведенных выше аналогов, была поставлена цель разработать «Telegram бота для оповещения студентов и преподавателей при ЧС», а так же «Веб-приложение для просмотра интерактивной карты и эмуляции при различных случаях ЧС» .

# Проектирование приложения

# Диаграмма вариантов использования

Проектирование — критический этап разработки программных решений, особенно в контексте управления чрезвычайными ситуациями. Начинающие разработчики часто пренебрегают этим этапом, полагаясь на интуицию или фрагментарные записи, что приводит к неорганизованности и повышению риска ошибок в критически важных системах. В условиях ЧС отсутствие четкого плана может стать фатальным, так как от скорости и точности работы системы зависят человеческие жизни.

Графическое представление архитектуры системы — ключевой инструмент проектирования. Вместо объемных текстовых описаний разработчики используют диаграммы, которые наглядно демонстрируют взаимодействие компонентов системы и ролей пользователей. Это упрощает коммуникацию в команде и снижает вероятность упущения важных функций.

Данная диаграмма определяет, какие функции системы доступны разным категориям пользователей. Модель строится на взаимодействии актеров (ролей) с вариантами использования — наборами действий, которые система предоставляет. Актером может быть человек, устройство или внешняя система, а вариант использования описывает цель взаимодействия без детализации реализации.

В разрабатываемом приложении должна быть разработана система аутентификации пользователей и предоставление прав на определенные действия в приложении. В ходе разработки программного средства будет реализовано два типа пользователей: обычный пользователь, кассир и администратор, имеющий расширенный функционал. Администратор помимо дополнительных возможностей, которые отличают его от обычного пользователя, может использовать также и функционал, доступный обычным пользователям.

Перед началом разработки необходимо определить роли, задачи, все варианты использования программного средства. Для этого необходимо построить диаграмму вариантов использования.

Для обеспечения безопасности пользователя с приложением была разработана эффективная система регистрации и аутентификации.

Особенности регистрации заключаются в том, что пользователь может легко создать учетную запись, заполнив простую форму с именем, телефонным номером и паролем. При этом пароль пользователя хэшируется и хранится в безопасной форме в базе данных MySQL, тем самым обеспечивая конфиденциальность информации. При входе в систему пользователь предоставляет свои данные, которые проверяются на соответствие с данными, хранящимся в базе данных. Аутентификация осуществляется с использованием bcrypt для проверс пароля на правильность. После успешной аутентификации пользователь получает доступ к функционалу приложения в зависимости от его роли. В зависимости от роли система предоставляет пользователю различные привилегии. Например, администраторы имеют доступ к определенным административным функциям, в то время как кассиры имеют доступ к функциям, связанным с обслуживанием клиентов. После успешной аутентификации пользователь сохраняется в сессии, что позволяет сохранить его состояние и обеспечивает безопасность во время сеанса использования приложения.

Для обеспечения безопасности маршрутов приложения используется middleware в методе checkAuth, который проверяет аутентификацию пользователя перед выполнением запроса. Если пользователь не авторизован, ему предлагается войти в систему через форму входа. Важно отметить, что пароли хранятся в зашифрованном виде, что гарантирует безопасность пользовательских данных в случае утечки информации. Эта система обеспечивает не только безопасность, но и удобство использования моего приложения, позволяя пользователям сосредоточиться на своих задачах, не беспокоясь о безопасности своих данных. Диаграмма вариантов использования для пользователя представлена на рисунке 2.1.

Изображение выглядит как диаграмма, линия, зарисовка, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.1 – Диаграмма вариантов использования для пользователя

Как видно из диаграммы вариантов использования пользователя, представленной выше, зарегистрированный пользователь. Однако незарегистрированный пользователь может взаимодействовать с системой, но только просматривать маршруты.

Зарегистрированный пользователь имеет 3 функциональных возможностей, а именно просмотра своего профиля, в котором пишется все забронированные заказы, редактировать его, удалить бронь на место. Также он может просмотреть список маршрут и забронировать место.

Диаграмма использования кассира представлена в приложении (потом нарисую). Следую из ей можно сделать вывод, что пользователь, обладающий такими правами, разрешается получить информацию об именах людей, которые забронировали место, для того чтобы отслеживать их прибытие на маршрут, также кассир имеет право заблокировать маршрут, после его отбытия от места стоянки. По сути своей кассир выполняет роль отслеживания за пассажирами и автомобилемв

# Алгоритм 1

фывфыв

# Архитектура приложения

Архитектура веб-приложений представляет собой в первую очередь организацию системы, состоящему из компонентов, их взаимосвязи меду собой и с окружающей средой.

В случаи разрабатываемого приложения для реализации веб-приложения было выбрано архитектура клиент-сервер. Клиент - сервер - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Эта архитектура позволяет распределить функционал и вычислительную нагрузку между клиентским и серверным приложением, обеспечивая тем самым эффективное взаимодействие между заказчиками, то есть клиентом и поставщиком услуг – организация. Сервер способен обслуживать несколько пользователей одновременно. В случаи если на сервер приходит несколько запросов одновременно, такие запросы выполняются сервером по очереди.

Клиент-серверная архитектура предлагает ряд преимуществ, включая:

* масштабируемость: Сервер может обслуживать большое количество клиентов одновременно, что позволяет масштабировать систему при увеличении обработки запросов.
* разделение обязанностей: Клиент и сервер выполняют разные задачи, что позволяет легко разделять ответственность между ними и облегчает сопровождение и модификацию системы.
* централизованное управление: Сервер является центральным узлом, который управляет данными и вычислениями, что упрощает администрирование системы.
* безопасность: Сервер может контролировать доступ к данным и ресурсам, обеспечивая безопасность системы.
* эффективная работа с данными: Клиенты могут обращаться к серверу для получения и обновления данных, что обеспечивает единообразие и целостность данных.
* использование различных технологий: Клиенты и серверы могут быть реализованы с использованием разных технологий, что позволяет выбирать наиболее подходящие средства для каждой из частей системы.

В целом, клиент-серверная архитектура обеспечивает эффективное взаимодействие между клиентскими приложениями и серверами, обеспечивая высокую производительность и удобство использования веб-приложений.

Архитектуру «клиент-сервер» принято разделять на три класса: одно-, двух- и трехуровневую. Однако, нельзя сказать, что в вопросе о таком разделении в сообществе ИТ-специалистов существует полный консенсус. Многие называют одноуровневую архитектуру двухуровневой и наоборот, то же можно сказать о соотношении двух- и трёхуровневой архитектур.[5]. Одноуровневая архитектура «клиент-сервер» (1-Tier) - такая, где все прикладные программы рассредоточены по рабочим станциям, которые обращаются к общему серверу баз данных или к общему файловому серверу. Никаких прикладных программ сервер при этом не исполняет, только предоставляет данные.

В целом, такая архитектура очень надежна, однако, ей сложно управлять, поскольку в каждой рабочей станции данные будут присутствовать в разных вариантах. Поэтому возникает проблема их синхронизации на отдельных машинах. В общем, как можно видеть из рисунка, в этой архитектуре просматривается еще один уровень - базы данных, что дает повод во многих случаях называть её двухуровневой.

К двухуровневой архитектуре «клиент-сервер» следует относить такую, в которой прикладные программы сосредоточены на сервере приложений (Application Server), например, сервере 1С или сервере CRM, а в рабочих станциях находятся программы-клиенты, которые предоставляют для пользователей интерфейс для работы с приложениями на общем сервере.

Такая архитектура представляется наиболее логичной для архитектуры «клиент-сервер». В ней, однако, можно выделить два варианта. Когда общие данные хранятся на сервере, а логика их обработки и бизнес-данные хранятся на клиентской машине, то такая архитектура носит название “fat client thin server” (толстый клиент, тонкий сервер). Когда не только данные, но и логика их обработки и бизнес-данные хранятся на сервере, то это называется “thin client fat server” (тонкий клиент, толстый сервер). Такая архитектура послужила прообразом облачных вычислений (Cloud Computing).

В трехуровневой архитектуре сервер баз данных, файловый сервер и другие представляют собой отдельный уровень, результаты работы которого использует сервер приложений. Логика данных и бизнес-логика находятся в сервере приложений. Все обращения клиентов к базе данных происходят через промежуточное программное обеспечение (middleware), которое находится на сервере приложений. Вследствие этого, повышается гибкость работы и производительность.

Трехуровневая архитектура веб-приложения представлена на рисунке 2.5

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.5 – Трехуровневая архитектура веб-приложения

Архитектура «клиент-сервер» - один из основных принципов работы сети Интернет. Любой веб-сайт, или приложение в Интернет работает на сервере, а его пользователи являются клиентами. Социальные сети (Фейсбук, ВК и пр.), сайты электронной коммерции (Amazon, Озон и др.) , мобильные приложения (Instagram и т.д.), устройства Интернета вещей (умные колонки или смарт-часы) работают на основе клиент-серверной архитектуры.

Хорошим примером работы системы «клиент-сервер» является автомобильный навигатор. Приложение навигации на сервере собирает данные с многих смартфонов пользователей, на которых установлены клиенты приложения. Кроме того, приложение навигации использует ещё и данные с сервера базы данных - геоинформационной системы, который предоставляет данные, например, о текущих ремонтах дорог, о появлении новых дорог и пр. Данные со многих клиентов (местоположение, скорость) обрабатывается сервером навигации и выдаётся на смартфоны пользователей в виде информации о средней скорости движения по тому или иному участку маршрута.

В целом в соответствии с клиентом-серверной архитектурой веб-приложение можно разделить на три отдельных компонента:

* клиентская часть, которая состоит из пользовательского интерфейса, с которым взаимодействуют пользователи;
* серверная часть, которая обрабатывает бизнес-логику веб-приложения и отвечает на запросы пользователя;

база данных – программное обеспечение на сервере, занимающиеся хранением данных, их обработкой и выдачей по запросу. База данных располагается на стороне сервера, а северная часть приложения обращается к базе данных, извлекает данные, которые необходимы для формирования страницы, запрошенной пользователем.

Архитектура приложения представлена на рисунке 2.6.

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.6 – Архитектура приложения

Разделение на клиент-серверную архитектуру позволяет упростить работу с приложением, поскольку настройкой каждого компонента можно заниматься отдельно и параллельно, при этом ускорив работу.

# Проектирование базы данных

фывфыв

# Выводы по разделу

Таким образом, была спроектирована база данных, описана UML-диаграмма, некоторые основные алгоритмы, а также определен основной стек технологий, используемый в данном проекте.

Каждый выбор, осуществленный в данной главе, является осознанным и обоснованным. Продумывание всех подобных моментов в начале, исключает неопределенность в будущем. Таким образом, заранее известен список всех используемых технологий, сферы деятельности, которые покрывают и некоторые основные алгоритмы, вокруг и на основе которых потом будут появляться новые механизмы взаимодействия между модулями системы. Таким образом, создавая администратора вначале, владелец сервиса освобождается от первичной инициализации базы данный администраторской информацией, необходимой для входа, что значительно упрощает весь процесс администрации.

В разделе 2.2 построена и описана схема архитектуры приложения. Также в данном разделе было рассмотрено проектирование базы данных. Проектирование осуществлялось при помощи построения модели данных, были определены таблицы и связи между ними. В разделе также были описаны таблицы с описанием каждого из столбцов и их ограничений целостности в спроектированной базе данных.

# Реализация веб-приложения

фывфыв

# Разработка серверной части приложения

фывфыв

# Подключение базы данных

фывфыв

# Реализация клиентской части приложения

фывфыв

# Выводы по разделу

фывфыв

# Анализ информационной безопасности приложения

фывфыв

# Выводы по разделу

фывфыв

# Тестирование приложения

фывфыв

# Руководство программиста

фывфывфыв

# Роль студента

фывфывфыв

# Роль преподавателя

фывфывфыв

# Выводы по разделу

фывфывфыв

# Технико-экономическое обоснование проекта

фывфыв

# Общая характеристика разрабатываемого программного средства

фвфывфыв

# Исходные данные и маркетинговый анализ

фывфывфыв

# Методика обоснования цены

фывфывфыв

# Определение объёма программного средства

фывфывфыв

# Основная заработная плата

фывфывфыв

# Дополнительная заработная плата

фывфывфыв

# Расчёт отчислений на социальные цели

фывфыв

# Расходы на материалы

фывфыв

# Расходы на специальное оборудование и платные услуги

фывфыв

# Расчет прочих прямых затрат

фывфыв

# Расчет общепроизводственных и общехозяйственных расходов

фывфыв

# Расчет суммы расходов на разработку программного средства

фывфыв

# Расходы на сопровождение и адаптацию

фывфыв

# Расчет общей суммы расходов

фывфыв

# Определение цены, оценка эффективности

фывфыв

# Выводы по разделу

фывфывфыв