Оглавление

[Введение 6](#_Toc194082357)

[1 Веб-приложение по усовершенствованию визуализации мониторинга качества воды 7](#_Toc194082358)

[1.1 Анализ предметной области 7](#_Toc194082359)

[1.2 Аналоги информационной системы 8](#_Toc194082360)

[1.3 Выводы и постановка задач на дипломное проектирование 12](#_Toc194082361)

[2 Разработка веб-приложения 14](#_Toc194082362)

[2.1 Программная структура веб-приложения 14](#_Toc194082363)

[2.2 Алгоритм работы программы 16](#_Toc194082364)

[2.3 Структура базы данных 19](#_Toc194082365)

[2.4 Выводы и оценка результатов разработки 20](#_Toc194082366)

[3 Анализ надежности веб-приложения 21](#_Toc194082367)

[3.1 Расчет надежности по модели сложности 21](#_Toc194082368)

[3.2 Расчет надежности по модели Джелинского – Моранды 25](#_Toc194082369)

[3.3 Расчет надежности по модели Муса 27](#_Toc194082370)

[4 Технико-экономическое обоснование разработки и использования веб-приложение по усовершенствованию визуализации мониторинга качества воды посредством введения цветных гексагонов на карте города минска 30](#_Toc194082371)

[4.1 Характеристика разработанного веб-приложения по индивидуальному заказу 30](#_Toc194082372)

[4.2 Расчет основных затрат на разработку 31](#_Toc194082373)

[4.3 Расчет результата от разработки и реализации веб-приложения 35](#_Toc194082382)

[4.4 Расчет показателей экономической эффективности разработки и использования веб-приложения 37](#_Toc194082383)

[7 Охрана труда. мероприятия по обеспечению высокой работоспособности и созданию комфортных условий труда при разработке и эксплуатации веб-приложения 41](#_Toc194082384)

[Заключение 45](#_Toc194082385)

# ВВЕДЕНИЕ

В современном мире качество питьевой воды остается одним из ключевых факторов, влияющих на здоровье населения. Контроль ее состава и своевременное информирование граждан о возможных отклонениях от нормы — важная задача для городских служб. В Минске данные о качестве воды публикуются на сайте Минскводоканала в табличном формате, что затрудняет их оперативный анализ и визуальное восприятие. Пользователям приходится вручную сравнивать значения с предельно допустимыми концентрациями (ПДК), что снижает удобство и скорость интерпретации данных.

Актуальность разработки обусловлена необходимостью улучшения способа визуализации информации о качестве воды для жителей Минска. Существующий табличный формат не позволяет быстро оценить общую ситуацию по городу, выявить районы с отклонениями от нормы или отследить динамику изменений. Внедрение интерактивной карты с цветовой дифференциацией показателей позволит сделать данные более наглядными и доступными для широкой аудитории.

Анализ текущего решения, предоставляемого сайтом Минсководоканала, показал, что табличный формат требует знаний в соответствующей области, что затрудняет понимаем о том, какое качество воды в заданном доме. Это создает нишу для разработки специализированного приложения, которое сможет улучшить наглядность данных.

Цель дипломного проектирования — разработка веб-приложения для усовершенствованной визуализации данных о качестве воды в Минске на основе гексагональной тепловой карты. Приложение должно автоматически загружать данные с сайта Минскводоканала, анализировать их на соответствие санитарным нормам и отображать результаты в формате, который не требует знаний норм предельно допустимых значений.

Объектом исследования являются методы визуализации пространственных данных, таких как качество воды в определённом районе, с помощью гексагональных сеток и тепловых карт. Предмет исследования — разработка алгоритмов обработки и отображения данных о качестве воды с учетом их динамики и территориального распределения.

Задачи проекта включают проектирование серверной части приложения, отвечающую за автоматизацию сбора и обработки данных с сайта Минскводоканала. разработка алгоритма классификации качества воды по районам города (алгоритм объединение группы домов в один гексагон соответствующего размера), оптимизацию запросов к *PostgreSQL*, реализация интерактивной карты с гексагональной сеткой и тестирование на реальных данных. Каждый из этапов разработки направлен на создание системы, которая будет не только функциональной, но и удобной в использовании.

Данный дипломный проект выполнен мной лично, проверен на заимствования, процент оригинальности составляет 72% (отчет о проверке на заимствования прилагается).

# 1 ВЕБ-ПРИЛОЖЕНИЕ ПО УСОВЕРШЕНСТВОВАНИЮ ВИЗУАЛИЗАЦИИ МОНИТОРИНГА КАЧЕСТВА ВОДЫ

## Анализ предметной области

В условиях роста урбанизации и увеличения нагрузки на городские инфраструктуры контроль качества питьевой воды становится одной из ключевых задач для обеспечения здоровья населения. Жители Минска, как и других крупных городов, нуждаются в достоверной и наглядной информации о состоянии водопроводной воды, чтобы своевременно реагировать на возможные отклонения от санитарных норм. В настоящее время данные о качестве воды публикуются на сайте Минскводоканала в табличном формате, что затрудняет их оперативный анализ и визуальное восприятие.

Разработка веб-приложения для усовершенствованной визуализации мониторинга качества воды направлена на решение этой проблемы. Приложение автоматически собирает данные с официальных источников, анализирует их на соответствие предельно допустимым концентрациям (ПДК) и отображает результаты в виде интерактивной тепловой карты с использованием гексагональной сетки. Такой подход позволяет пользователям быстро оценить ситуацию по всему городу, выявить районы с потенциальными рисками и отслеживать динамику изменений.

Для разработки такого веб-приложения можно выделить основные концептуальные требования:

1 Автоматизированный сбор данных: получение актуальных показателей качества воды с сайта Минскводоканала.

2 Анализ и классификация данных: сравнение значений с ПДК и присвоение цветовых меток в зависимости от уровня отклонения.

3 Визуализация данных на карте: отображение данных в виде гексагональной сетки, обеспечивающей равномерное покрытие территории и плавные цветовые переходы.

4 Исторический анализ: хранение и отображение данных за предыдущие периоды для выявления проблемных мест.

Целевая аудитория такого веб-приложения включает жителей Минска, которые смогут быстро оценить качество воды в своем районе и принять меры при необходимости, экологическими организации и городские службы, которые смогут оперативно выявлять проблемные зоны.

Система состоит из трех основных структурных частей: веб-интерфейса, сервера для обработки собираемой информации и базы данных. Веб-интерфейс служит точкой входа в систему. Такой веб-приложение позволяет просматривать информацию о качестве воды не только в своей доме, но и в целом по городу Минску, позволяет сравнивать два временных промежутка времени между собой. Серверная часть системы отвечает за обработку данных, выполнение бизнес-логики и управление взаимодействием между веб-интерфейсом и базой данных. Она включает API для приема и отправки данных, алгоритмы расчета цвета для гексагона и сохранении информации в базу данных. *PostgreSQL* – выбранная база данных, которая обеспечивает хранение информации о гексагонах, времени расчета, параметрах воды на определенный момент времени.

Взаимодействие между модулями организовано следующим образом: Сервер по крону собирает данные о качестве воды всех домов города Минска, после чего собирает преобразованную информацию в базу данных. Веб-интерфейс может запрашивать данные как по улице, так и по рассчитанному гексагону, что позволяет поддержать как нативный интерфейс (табличный формат), так и более продвинутое решение в виде гексагональной разметки.

Таким образом, система обеспечивает наглядное представление информации о качестве воды, упрощает ее анализ и способствует повышению осведомленности населения.

## 1.2 Аналоги информационной системы

Анализ существующих решений показал, что на рынке представлено несколько категорий систем, которые частично покрывают функционал. Однако ни одно из них не предлагает комплексного подхода, сочетающего и актуальные данные о качеств воды, и интеграцию с картами и тепловую карту. Рассмотрим наиболее значимые аналоги.

Сайт Минского водоканала — на сайте представлена информцию о том, какие параметры воды сейчас в конкретном доме. Расписаны технолгии, по которым собираются данные, как часто эти данные обновляются, какие метрики используются для измерения параметров. Также предоставлен интерфейс для взаимодействия Яндекс Картами: можно выбрать конкретный дом, а не вводить название улицы и дома в поисковой строке. Также на сайте представлена различная информация, связанная с истониками воды в городе Минске.

Основные достоинства включают простоту использования, поддержку не только веб-интерфейса для ПК, но и ддля мобильных устройств, доступность на белорусском рынке и слабовидящих людей. Интерфейс сайта интуитивно понятен, что позволяет пользователям быстро освоить его без дополнительного обучения. Однако, несмотря на свои преимущества, система не адаптирована для пользователей, которые хотят быстро увидеть ситуацию не только в своем доме, но и в соседних районах – необходимо вводить каждую улицу и дом по отдельности, что замедляет анализ текущей ситуации.

Интерфейс приложения Toggl представлен на рисунке 1.1.

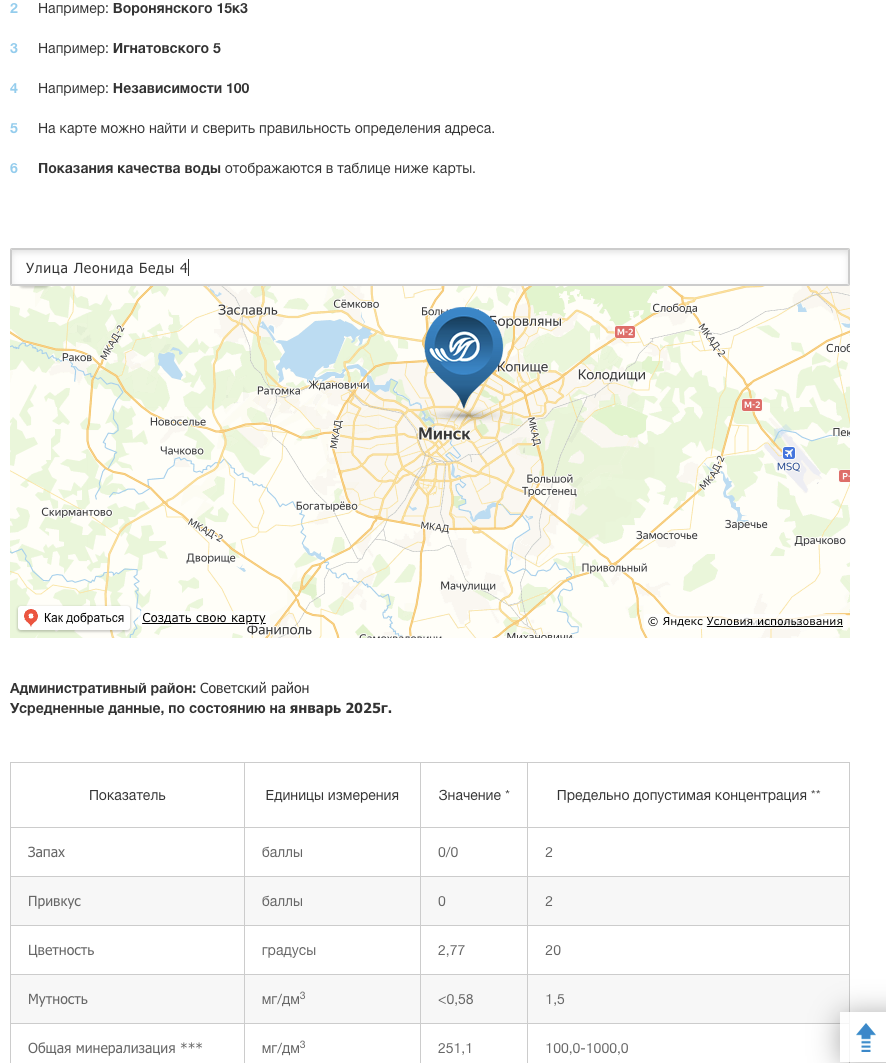


Рисунок 1.1 – Интерфейс сайта минского водоканала

Water Quality Map — это сайт, который отображает информацию по различным показателям воды в реках, каналах, озерах, родниках, колодцах, скважин и водохранилищах в различных уголках России. Преимущества такого сайта – это отметки на физической карте мира конкретных водохранилищ, что позволяет смотреть на ситуацию о качестве воды по более чем 15 параметрам в различных районах. Однако у сайта есть и недостатки, нет понимания того, что происходит в более мелких районах, а не только в водохранилищах, что может привести к пропуску какой-нибудь опасной зоны, в которой качество воды, например, могло ухудшится из-за техногенной катастрофы. Также минусом является неактуальность данных на этом сайте. По датам некоторых водохранилищ, информация не обновлялась уже более чем 3 года.

Для жителей Минска на таком сайте, к сожалению, нет полезной информации, так как ближайшее водохранилище находится довольно далеко от границ Беларуси.

Сайт также предоставляет возможность присылать данные из лабораторий или других аккредитованных организаций о качестве воды в определенном районе или водохранилище, что позволяет собирать данные не только от выбранного списка лабораторий, но еще и с пользователей, которые в этом заинтересованы. Так, например, если ранее эти карты не поддерживали белорусский регион, то теоретически такая поддержка возможна благодаря такому функционалу.

Интерфейс сайта представлен на рисунке 1.2

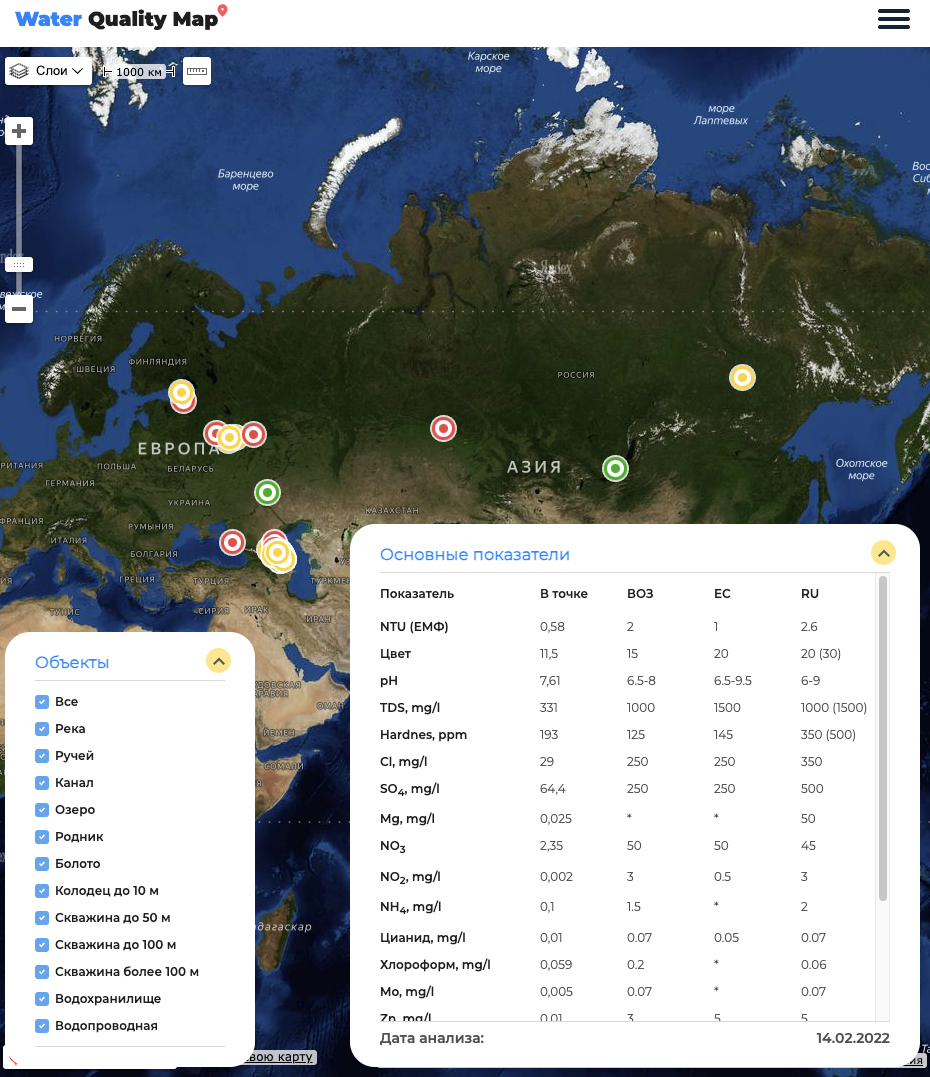


Рисунок 1.2 – Интерфейс сайта WaterQualityMap

Wasserkiez — это веб-приложение для визуализации водяных контуров внутри стран европы. На карте отмечены водохранилища, пункты выдачи воды. Проект разработан поверх базы данных немецкой компании tip:tap, которая занимается производством питьевой воды в Европе.

Этот сайт не подходит для анализа большого количества параметров воды, так как он предоставляет информацию только о местах, связанных с предоставлением питьевой воды. Интерфейс сайта приведен на рисунке 1.3.

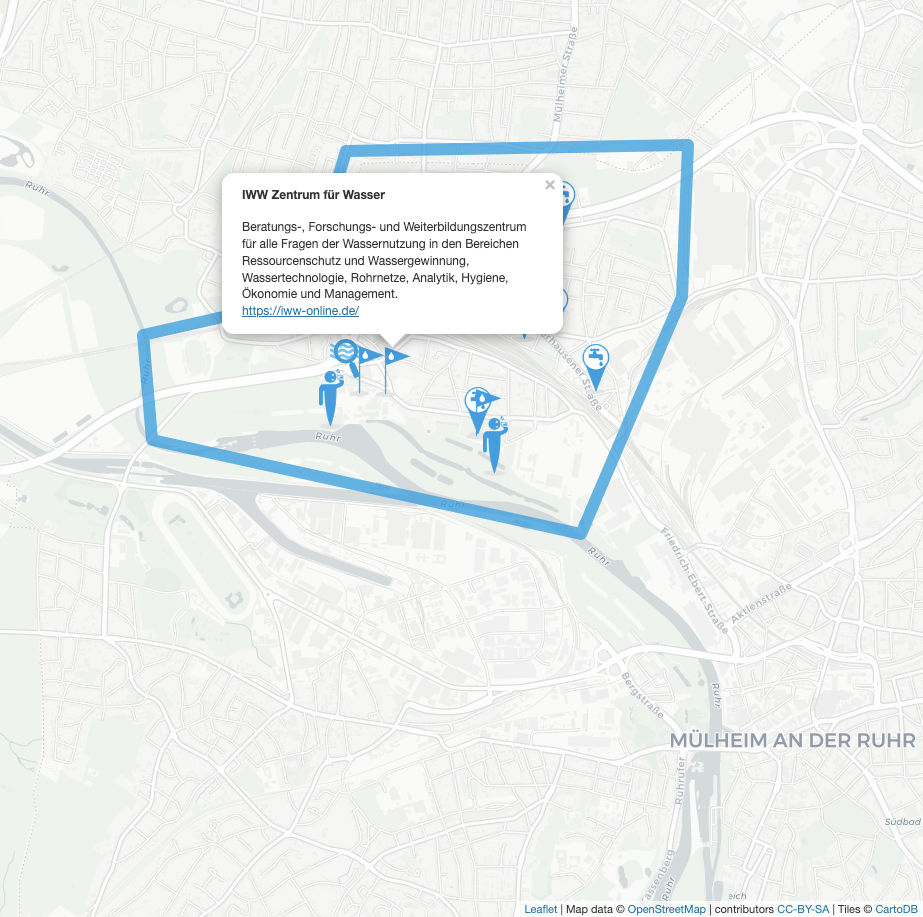


Рисунок 1.3 – Интерфейс сайта Wasserkiez

Uber H3 Viewer — это веб-приложение, которое позволяет по номеру гексагона и его резолюции, т.е. его размера, узнать его местоположение на карте. Имеет достаточно удобный и понятный интерфейс, но, к сожалению, не предоставляет публичного API для взаимодействия. Интерфейс сайта с гексагональным покрытием карты представлен на рисунке 1.4.



Рисунок 1.4 – Интерфейса сайта Uber H3 Viwer

Итог анализа аналогов показывает, что существующие системы частично покрывают необходимый функционал. Однако ни одно из решений не предлагает комплексного подхода, сочетающего и агрегацию данных о качестве воды, разметку на карте с помощью тепловой визуализации и интеграцию с картами. Все приведенные аналоги являются бесплатными решениями, а некоторые из них даже находятся в открытом доступе с исходным кодом под лицензией *MIT*. Разрабатываемая система будет выгодно отличаться доступностью для белорусского рынка (в частности для жителей города Минска), низкой стоимостью внедрения и специализированным функционалом для сравнения параметров различных временных периодов.

## 1.3 Выводы и постановка задач на дипломное проектирование

Разработка веб-приложения для усовершенствованного мониторинга качества воды в г. Минске направлена на решение проблемы неудобного представления данных о состоянии водопроводной воды, которые в настоящее время публикуются в табличном формате на сайте Минскводоканала. Основная цель системы – обеспечить наглядную и интерактивную визуализацию данных о качестве воды с использованием гексагональной сетки на карте города, что позволит пользователям оперативно оценивать ситуацию, выявлять проблемные зоны и отслеживать динамику изменений.

Требования к разрабатываемой системе включают создание трех основных компонентов: веб-интерфейса приложения, серверной части для обработки данных и базы данных для хранения информации. Веб-приложение обеспечивает интеграцию с картами для визуализации гексагонов и качества. Воды, а также за возможность сравнения двух временных периодов. Серверная часть, реализованная на Python, отвечает за обработку данных, выполнение бизнес-логики и управление взаимодействием между веб-интерфейсом и базой данных. База данных на PostgreSQL обеспечивает надежное хранение информации о поездках, пользователях и результатах аналитики.

Обмен данными между веб-интерфейсом и сервером осуществляется через REST API, что обеспечивает гибкость и масштабируемость системы.

Взаимодействие со смежными системами включает интеграцию с картографическими сервисами, такими как *Google Maps* или *Yandex Maps*. Также возможная интеграция с технологиями Uber, которые позволяют вычислять корректную разметку гексагонов на карте. В перспективе система может быть расширена за счет добавления оповещения пользователей о плохом качестве воды в их районе, для чего нужно будет поддержать систему пользователей, а также поддержать возможность отправлять уведомления на телефон и(или) почту.

Эргономика системы играет важную роль в ее успешном внедрении. Интерфейс приложения должен быть интуитивно понятным и удобным для использования гражданами Минска, а серверная часть — обеспечивать быстрый доступ к данным и формирование карты в удобном для пользователей формате.

Разрабатываемая система обладает рядом преимуществ. Во-первых, она предлагает специализированный функционал для анализа различных временных промежутках. Во-вторых, использование современных технологий, таких как Python, JavaScript, и PostgreSQL, позволяет минимизировать затраты на разработку и поддержку. В-третьих, модульная архитектура системы обеспечивает ее масштабируемость, что позволяет легко расширять функционал в будущем. Наконец, система разрабатывается с учетом потребностей белорусского рынка.

Выбор технологий для разработки системы обоснован их надежностью, производительностью и доступностью. Python обеспечивает высокую достаточную для целей веб-приложения производительность, быстроту разработки и безопасность серверной части, JavaScript, в связке с большим количеством фреймворков, позволяет создать удобный пользовательский интерфейс, а PostgreSQL — надежную и масштабируемую базу данных.

Таким образом, разрабатываемая система представляет собой специализированное решение, которое сочетает в себе простоту использования, аналитику текущей ситуации по городу, а также быстрый доступ к информации.

# 2 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ

## 2.1 Программная структура веб-приложения

Разработка веб-приложения для визуализации мониторинга качества воды потребовала тщательного выбора современных технологий, обеспечивающих высокую производительность, удобство взаимодействия с пользователем и надежность хранения данных. Система построена по трехзвенной архитектуре, включающей фронтенд, бэкенд и базу данных, каждая часть которой реализована с использованием оптимальных технологических решений.

Для разработки пользовательского интерфейса был выбран React.js – современный JavaScript-фреймворк. React.js – это богатая экосистема дополнительных библиотек и инструментов, а также уже не первый год существование на рынке, что дает свои плюсы – большое количество материалов на просторах интернета. Фронтенд часть системы также использует Leaflet.js для интерактивного взаимодействия с картами. С помощью такой библиотеки можно выбрать в качестве основного слоя любую из доступных карт (будь то Google Maps, Yandex Maps или Open Street Maps), на который можно добавляет различные объекты. Например, такими объектами могут быть пины на карте или, как в нашем случае, гексагоны с цветовой разметкой. Это фреймворк, который предоставляет набор функций разработана на языке программирования Java с использованием фреймворка Spring Boot. Также используется библиотека D3 для создания гексагонов на карте.

Серверная часть приложения разработана на Python с фреймворком Django для взаимодействия по REST API. Также среди используемых библиотек есть такие, как Selenium, для возможности обрабатывать данные с сайтов минского водоканала, а также psycopg2 для взаимодействия с PostgreSQL. Серверная часть также отвечает за подсчет средних значений по гексагону. Эта логика специально отселена в отдельную компоненту – крон таску, которая рассчитывает значения для всех гексагонов и которая реализована с помощью apsheduler. ApSheduler – это технология, которая позволяет периодически запускать действие, указанное в специальных функциях. Так, например, в нашем случае мы запускаем периодически тяжелые расчеты (сбор информации о всех улицах города Минска и параметрах воды в домах), после чего сохраняем всю полезную информацию внутрь базы данных. Тут стоит отметить, что из-за асинхронной схемы взаимодействия между частью с расчетами и потенциальными клиентами сервиса могут возникать повторные расчеты – если выполнение задачи по крону еще не завершено, а пользователь уже хочет получить данные.

Для хранения данных была выбрана реляционная база данных PostgreSQL. Этот выбор обусловлен ее надежностью, поддержкой сложных запросов и масштабируемостью. PostgreSQL позволяет эффективно хранить и обрабатывать большие объемы данных.

В процессе разработки использовались следующие инструменты: PyCharm, WebStorm и Visual Studio Code в качестве основных сред разработки для написания кода на Python и JS, Git для контроля версий и эффективного управления кодом.

Выбор технологий для разработки системы был основан на их надежности, доступности и скорости разработки. Python и JS обеспечили доступность, легкость в освоении и скорость разработки backend-части, а PostgreSQL — надежное хранение данных. Таким образом, выбранные технологии и инструменты обеспечили эффективную разработку системы, которая соответствует всем поставленным требованиям и готова к дальнейшему масштабированию.

Система веб-приложения для визуализации качества воды строится вокруг центрального модуля обработки и представления данных. Ядром системы выступает серверная часть, реализованная на Python с использованием фреймворка *Django*, который обеспечивает гибкость и простоту разработки API. Сервер взаимодействует с базой данных PostgreSQL, где хранятся как текущие показатели качества воды, так и исторические данные для анализа тенденций. Для обеспечения бесперебойной работы системы реализован механизм автоматического обновления данных через регулярные запросы к API Минскводоканала.

Система состоит из нескольких модулей, которые взаимодействуют между собой через REST API. Это обеспечивает масштабируемость и модульность системы, позволяя легко добавлять новые функции или изменять существующие без необходимости переписывания всей системы. Основные модули включают:

1 Веб-интерфейс, предназначенный для пользователей, позволяющий взаимодействовать с картой и вводить адреса интересующих их мест. Пользователь может указать как улицу, так и номер интересующего гексагона. Также пользователь может сравнивать две даты по качеству воды.

2 Модуль обработки данных и формирования агрегируемых параметров по гексагону (серверная часть) отвечающий за обработку данных, полученных из минского водоканала. Сервер рассчитывает средние показатели качества воды в гексагонах и сохраняет такую информацию в PostgreSQL. Сервер делает это в асинхронном режиме в качестве крон задачи, что позволяет реализовать более эффективный расчет данных на карте.

3 Модуль хранения данных (база данных) обеспечивает хранение всех данных, включая информацию о гексагонах, параметрах воды и результатах анализа. База данных построена на PostgreSQL, что обеспечивает надежность и масштабируемость.

Взаимодействие между модулями организовано следующим образом: сервер в асинхронном режиме рассчитывает полезные для всей системы данные – это параметры воды по гексагонам. Для выполнения этой цели серверу необходимо получить список всех улиц и всех домов города Минска, после чего рассчитать средние показатели качества воды в конкретном гексагоне. Веб-интерфейс отправляет данные в сервер для извлечения информации из PostgreSQL, которая предварительно была посчитана с помощью асинхронной задачи. При отсутствии полезной информации в PostgreSQL (например, если задача была временно отключена или выполняется слишком долго) происходит синхронной расчет информации, запрошенной пользователем.

Пользователи взаимодействуют с системой через интерактивную карту, где могут кликать на цветные гексагоны для просмотра детальных показателей качества воды во всплывающих окнах. В интерфейсе есть настройки отображения данных по временным периодам. Пользователи могут масштабировать карту для детального изучения конкретных районов или просматривать общую картину по всему городу. Дополнительная панель инструментов позволяет переключаться между различными режимами визуализации, включая тепловую карту и сравнительный анализ по месяцам. Все данные обновляются автоматически при изменении информации в источнике, обеспечивая актуальность представленной информации.

Таким образом, система обеспечивает комплексный подход визуализации качества воды. Взаимодействие между модулями через REST API обеспечивает гибкость и масштабируемость системы, что позволяет легко адаптировать ее под нужды конкретных пользователей или организаций.

## 2.2 Алгоритм работы программы

Алгоритм работы программы включает несколько ключевых этапов, начиная с входа пользователя на главную страницу веб-приложения и заканчивая агрегацией результатов расчетов и сохранением их в базу данных.

Ядром приложения выступает схема асинхронного расчета и агрегации параметров воды для всех гексагонов города Минска. Во время выполнения задачи, сервер запрашивает данные из внешних источников для получения всех домов города Минска, а также для получения параметров воды по адресам.

Рассмотрим алгоритм, по которому работает асинхронная задача:

1 Сервер по истечению определенного *ttl* запускает задачу для расчета параметров.

2 После запуска сервер собирает данные о всех домах города Минска.

3 Полученные адреса с помощью API карт превращаются в координаты, которые состоят из широты и долготы.

4 После формирования координат такой адрес превращается в гексагон с помощью соответствующей библиотеки h3.

5 Для всех адресов, полученных в пункте 2, запрашивается информация о параметрах воды с сайта минского водоканала.

6 Расчет параметров и их агрегация выполняется на последнем этапе, после чего результаты сохраняются в базу данных.

Для наилучшего описания алгоритма действий всех участников системы была выбрана диаграмма взаимодействия, которая представлена на рисунке 2.1. Данные диаграммы используются, чтобы продемонстрировать взаимодействие пользователей во времени и выделить этапы работы системы.

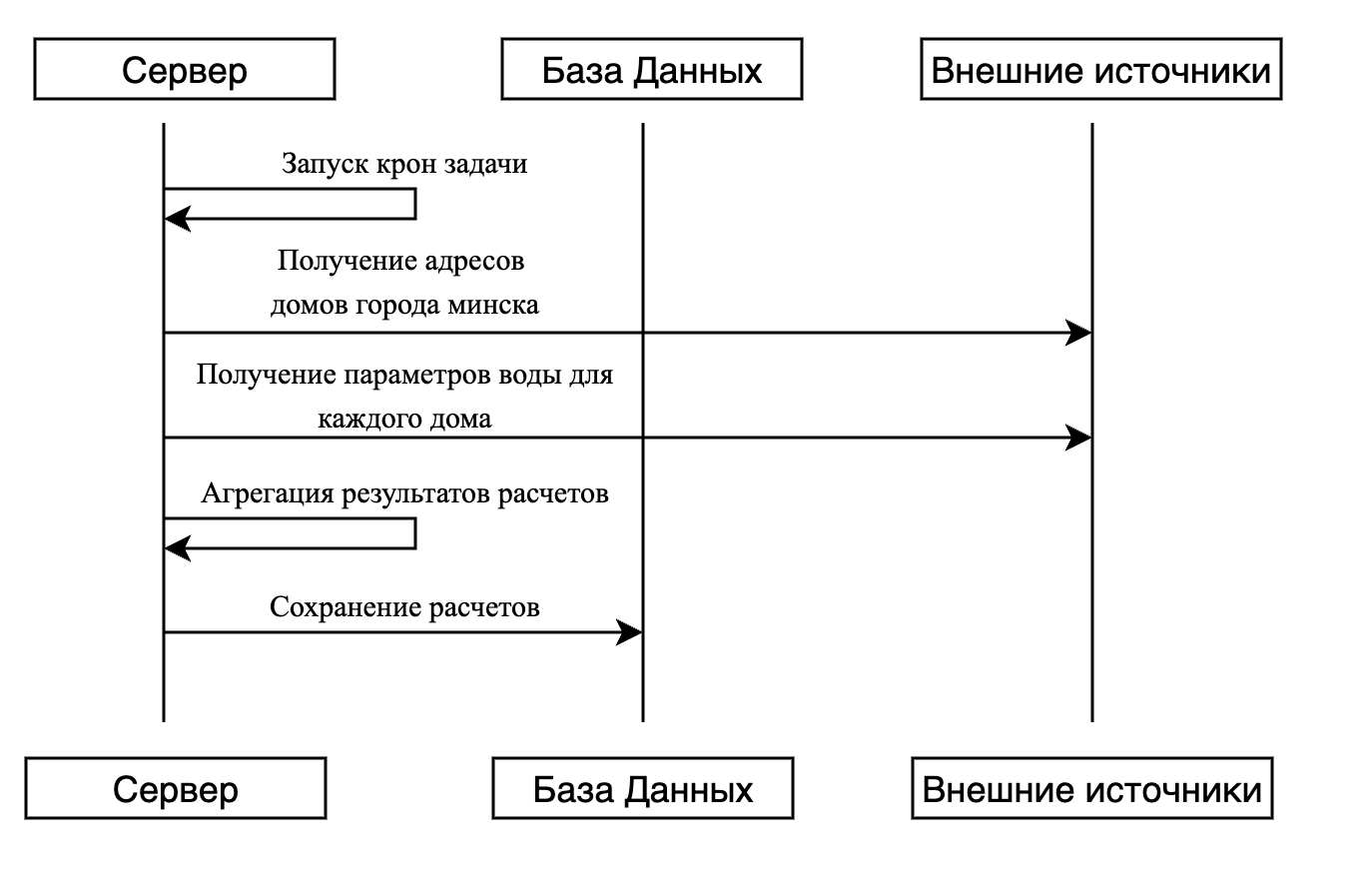


Рисунок 2.1 – Диаграмма взаимодействия при расчете параметров воды всех гексагонов города Минска

Также хочется поддержать возможность пользователя получать параметры воды для конкретного адреса в том же веб-приложении, при условии того что данные из асинхронной задачи еще не были посчитаны:

1 Пользователь вводит название улицы и номер дома.

2 Веб-интерфейс отправляет запрос серверу для получения параметров для адреса, введенного пользователем.

3 На стороне сервера происходит декодирование адреса в гексагон и проверка наличия параметров в базе данных

4 При отсутствии значений происходит синхронный расчет параметров воды.

5 Рассчитанные параметры передаются веб-интерфейсу.

6 Для пользователя формируется необходимый интерфейс, на котором отображается гексагон, который принадлежит адресу и отображаются параметры воды.

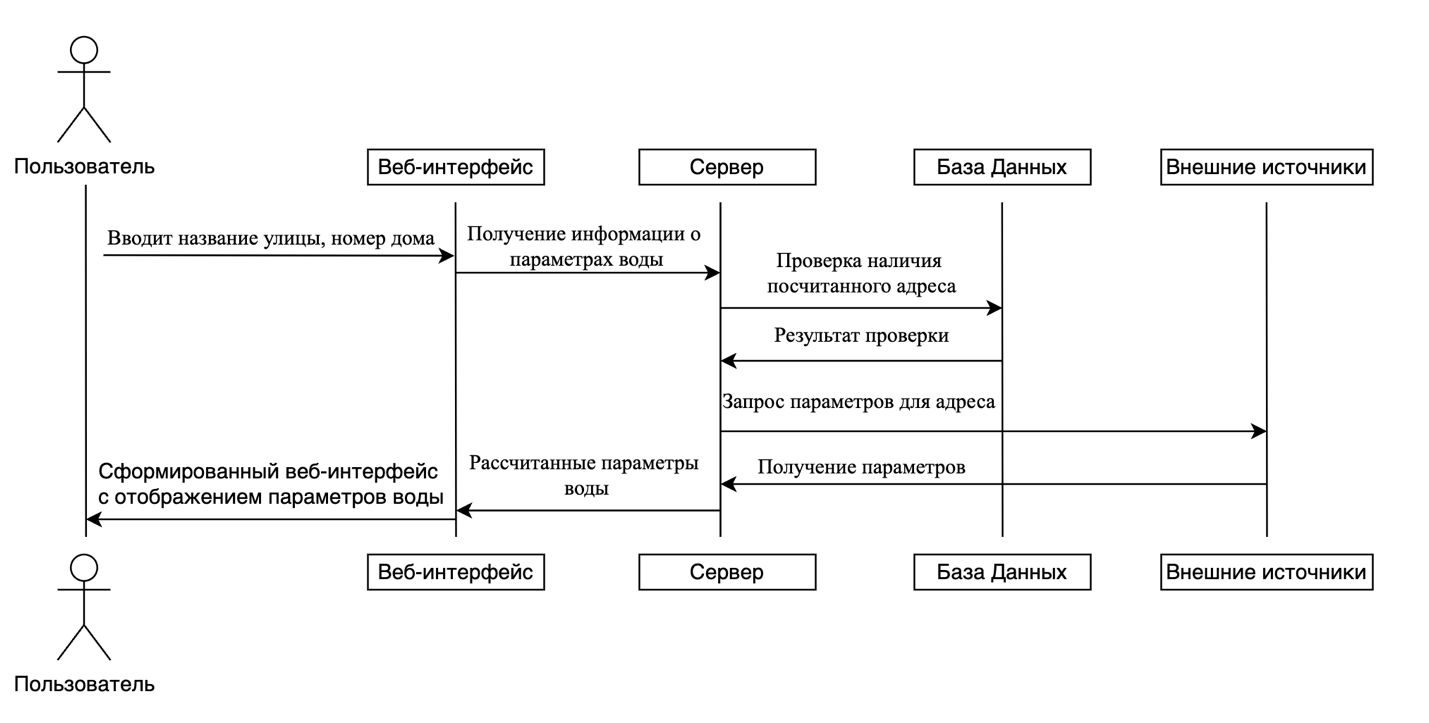


Рисунок 2.2 – Диаграмма взаимодействия пользователя с веб-приложением при отсутствии параметров адреса внутри базы данных

Также на стороне сервера присутствует алгоритм декодирования адреса в необходимые для веб-приложения сущности: гексагон (*hex\_id*) и координаты. Алгоритм:

1 Полученный адрес с помощью API карт преобразуется в координаты, которые содержат широту и долготу.

2 Из интерфейса веб-приложения извлекается информация о текущей резолюции гексагонов. Резолюция гексагона – коэффициент размера области, которую покрывает гексагон. Формирование гексагонов происходит посредством библиотеки h3, которую разработал *Uber*.

3 Формируется сущность *Address*, которая далее используется в системе как единый интерфейс обращения с координатами, названием адреса в привычном для человека формате и hex\_id.

Немаловажную часть также занимает алгоритм агрегации результатов вычисления параметров воды для каждого отдельного адреса в соответствующие гексагоны. Алгоритм объединения гексагонов и их параметров:

1 Среди всех доступных для сервера адресов делаем следующую операцию: перевод человекочитаемого адреса в сущность *Address*, алгоритм формирования которой был описан выше.

2 Объединяем сущности Address в группы, где находится одинаковый hex\_id.

3 По каждому из доступных параметров находим среднее значение между всеми адресами, вошедшими в группу.

Алгоритм формирования цвета гексагона был выбран следующий:

1 Нормализуем значения параметров воды согласно ПДК.

2 После нормализации находим максимальное отклонение.

3 Преобразуем отклонение в цвет: для этого используем формат HSV (hue, saturation, value), где первый параметр – это цвет, saturation – насыщенность и value – яркость. Для того чтобы преобразовать отклонение в цвет необходимо задать два граничных значения – это красный и зеленый цвета, а также зафиксировать яркость и насыщенность.

4 Преобразуем HSV в RGB.

Таким образом алгоритм работы программы отображает основные этапы спроектированной системы. Система может как использовать данные, посчитанные с помощью асинхронного процесса, так и запрашивать данные с помощью точечного адреса. Также в этом разделе был отражен алгоритм перевода адресов в гексагоны и координаты и продемонстрирован алгоритм конвертации параметров воды в цветовую схему.

## 2.3 Структура базы данных

База данных веб-приложения для усовершенствования мониторинга качества воды состоит из таблицы hexagons, в которой хранится информация о гексагонах.

Таблица одна и имеет следующую схему: поле hex\_id, которое является главным для поиска, created\_at – timestamp, который отражает момент попадания гексагона в систему для расчета его параметров воды. Такое поле необходимо, чтобы фильтровать данные, которые оказались либо слишком старыми, для текущего промежутка времени, либо слишком новыми, если сейчас пользователь находится в режиме сравнения. Также в таблице присутствует еще два поле: hex\_color и water\_parameters. Где hex\_color – это строка, которая отвечает за то, какой цвет сейчас должен быть у гексагона в шестнадцатеричном формате. Поле water\_parameters содержит все возможные параметры воды для текущего гексагона. Например, запах, жесткость, минерализация.

Для дальнейших доработок, например, при введении уведомлений пользователем через их личный аккаунт, могут быть введение еще несколько соответствующих таблиц. Одной из таких таблиц может быть users, в которой будут хранится email’ы, по которым необходимо отправлять уведомления о плохом качестве воды в их районе.

Таким образом, база данных системы обеспечивает гибкость и масштабируемость, позволяя эффективно хранить и обрабатывать данные о гексагонах, времени их создания, а также параметрах воды. Такая структура позволяет системе анализировать средние показатели и выбирать временные промежутки для отображения.

## 2.4 Выводы и оценка результатов разработки

Разработка веб-приложения для визуализации мониторинга качества воды успешно завершена. Система полностью соответствует поставленным требованиям и обеспечивает эффективное решение задач сбора, анализа и наглядного представления данных о состоянии водных ресурсов.

Ключевым преимуществом приложения является его специализированный функционал, ориентированный на визуализацию качества воды через гексагональную сетку. Система автоматически загружает и анализирует данные с сайта Минскводоканала, преобразуя табличные показатели в интуитивно понятную цветовую карту. Пользователи могут изучать детализированную информацию по районам, применять фильтры для анализа конкретных параметров воды и сравнивать данные за разные периоды.

В процессе разработки были успешно решены задачи интеграции с внешними источниками данных, проектирования алгоритмов классификации показателей воды и реализации интерактивной карты на основе Leaflet.js и D3.js. Система прошла тестирование на реальных данных, подтвердив точность визуализации и стабильность работы.

Таким образом, разработанное веб-приложение представляет собой эффективный инструмент для мониторинга качества воды, сочетающий удобство использования с мощной аналитической составляющей.

# 3 АНАЛИЗ НАДЕЖНОСТИ ВЕБ-ПРИЛОЖЕНИЯ

## 3.1 Расчет надежности по модели сложности

Модель сложности программного обеспечения основывается на предположении, что уровень безошибочности продукта может быть предсказан с использованием метрик сложности. Это справедливо для непреднамеренных уязвимостей, так как, чем сложнее и больше программа, тем выше вероятность ошибок при её написании и модификации. Для расчета надежности программного средства по модели сложности используются метрики размера, сложности потока управления, сложности потока данных, а также объектно-ориентированные метрики.

К метрикам размера относят объём (*V*) и потенциальный объём (*V\**) программного средства.

Объём программного средства *V* рассчитывают по формуле (3.1):

|  |  |
| --- | --- |
|  | (3.1) |

где:

– n1 =15 (число уникальных операторов);

– n2 =30 (число уникальных операндов);

– N1 =170 (общее число операторов);

– N2 =85 (общее число операндов).

Потенциальный объем программы определяется по формуле 3.2:

(3.2)

,

где *n*\* – теоретический словарь программы (словарный запас, необходимый для написания программы с учетом того, что необходимая функция уже реализована в языке программирования). Теоретический словарь программы равен сумме количества операторов и операндов программы, при условии того, что нужная функциональность уже реализована в программе. Для разрабатываемого программного средства *n*\*= 136.

Для разрабатываемого программного средства значения объема и потенциального объема равны:

.

Для определения метрик сложности потока управления используются следующие характеристики:

– CL – абсолютная сложность программы, характеризующаяся количеством операторов условия;

– cl – относительная сложность программы, определяемая как отношение CL к общему числу операторов;

– CLI – максимальный уровень вложенности операторов условия.

Для разрабатываемого программного средства значения этих характеристик равны: CL =20, cl = 26, CLI = 42.

Метрика сложности потока данных, предложенная Чепиным, оценивает информационную прочность программного модуля на основе анализа использования переменных из списка ввода-вывода. Все переменные делятся на четыре группы:

– P (вводимые переменные для расчетов и обеспечения вывода);

– M (модифицируемые или создаваемые внутри программы переменные);

– C (переменные, участвующие в управлении работой программного модуля);

– T (не используемые в программе переменные).

Метрика Чепина рассчитывается по формуле 3.3:

(3.3)

Для разрабатываемого программного средства значение метрики Чепина равно:

Теоретическая длина программы N рассчитывается по формуле 3.4:

(3.4)

где n1 – словарь операторов, n2– словарь операндов.

Для разрабатываемого программного средства = 205.

Уровень качества программирования L определяется как отношение потенциального объема к фактическому по формуле 3.5:

(3.5)

Для разрабатываемого программного средства L = 0,69.

Аппроксимированный уровень качества программирования L^ рассчитывается по формуле 3.6:

(3.6)

​

Для разрабатываемого программного средства = 0,05.

Интеллектуальные усилия на разработку программы E рассчитываются по формуле 3.7:

Для разрабатываемого программного средства E = 2035,8

(3.7)

Для оценки объектно-ориентированных характеристик программного средства используются метрики Чидамбера и Кемерера:

– WMC – суммарная сложность всех методов класса;

– DIT – глубина дерева наследования;

– NOC – количество потомков;

– CBO – сцепление между классами;

– RFC – отклик для класса;

– LCOM – недостаток сцепления методов.

Для разрабатываемого программного средства значения этих метрик равны: WMC = 201, DIT = 2, NOC = 3, CBO = 7, RFC = 20, LCOM = 20.

Результаты расчетов представлены в таблице 3.1.

Таблица 3.1 – Результаты расчетов

|  |  |  |  |
| --- | --- | --- | --- |
| Метрики ПС | amin | amax | ai |
| V | 108 | 1364 | 1400,4 |
| V\* | 540 | 83362,41 | 963,9 |
| CL | 7 | 368 | 20 |
| Cl | 25 | 1278 | 26 |
| CLI | 41 | 4213 | 42 |
| Q | 67 | 3589 | 170 |
| N^ | 6 | 186 | 205 |
| L | 0,0054 | 2 | 0,69 |
| L^ | 0 | 3 | 0,05 |
| E | 29 | 967 | 2035,83 |
| WMC | 13 | 299 | 201 |
| DIT | 1 | 8 | 2 |
| NOC | 1 | 32 | 3 |
| CBO | 1 | 27 | 7 |
| RFC | 1 | 163 | 20 |
| LCOM | –39 | 387 | 20 |

Исходя из полученных метрик рассчитаны нормативный и фактический уровни каждой метрики (формулы (3.8) и (3.9) соответственно):

(3.8)

(3.9)

По полученным результатам определен дискриминант каждой метрики, используя формулу (3.10):

(3.10)

Результаты вычислений по формулам (3.8 – 3.10) представлены в таблице 3.2.

Таблица 4.2 – Результаты вычисления

|  |  |  |  |
| --- | --- | --- | --- |
| Метрики ПС | xmin | xфi | di |
| V | 0,07917888563 | 1,026686217 | -0,00223503021 |
| V\* | 0,006477739787 | 0,01156276552 | 0,5573567619 |
| CL | 0,01902173913 | 0,05434782609 | 0,3373961219 |
| Cl | 0,01956181534 | 0,02034428795 | 0,9607710725 |
| CLI | 0,009731782578 | 0,009969143128 | 0,97595649 |
| Q | 0,01866815269 | 0,04736695458 | 0,3825917761 |
| N^ | 0,03225806452 | 1,102150538 | -0,003089430894 |
| L | 0,0027 | 0,345 | 0,005139964862 |
| L^ | 0 | 0,01666666667 | 0 |
| E | 0,02998965874 | 2,105305067 | -0,01623163562 |
| WMC | 0,04347826087 | 0,6722408027 | 0,02216191768 |
| DIT | 0,125 | 0,25 | 0,4285714286 |
| NOC | 0,03125 | 0,09375 | 0,311827957 |
| CBO | 0,03703703704 | 0,2592592593 | 0,1098901099 |
| RFC | 0,006134969325 | 0,1226993865 | 0,04413580247 |
| LCOM | -0,1007751938 | 0,05167958656 | -1,679929577 |

Риск снижения надежности рассчитывается по формуле 3.11:

(3.11)

Где λi – весовые коэффициенты для каждой метрики. Для упрощения расчетов можно считать, что все метрики вносят равный вклад в надежность, т.е. λi = 1/16.

Для разрабатываемого программного средства вероятность безотказной работы равна 3.12:

(3.12)

Таким образом, вероятность безотказной работы программного средства по модели сложности составляет приблизительно 0,6.

## 3**.2 Расчет надежности по модели Джелинского – Моранды**

Модель Джелинского – Моранды используется для оценки надежности программного обеспечения на основе данных о времени между обнаружением ошибок Такая модель строится на основе следующих допущений:

1 Интенсивность обнаружения ошибок λ(t) пропорциональна текущему числу ошибок в программном средстве, т. е. числу оставшихся ошибок.

2 Все ошибки одинаково вероятны, и их появления независимы.

3 Каждая ошибка имеет один и тот же порядок серьезности.

4 Время до следующего отказа (ошибки) распределено экспоненциально.

5 Программное средство функционирует в среде, близкой к реальной.

6 Ошибки постоянно корректируются без внесения в программное средство новых.

7 λ(t) = const в интервале между двумя соседними ошибками.

В данном разделе проведем расчет надежности программного средства на основе модели Джелинского – Моранды. Общее время тестирования программы составило 17 дней. За это время было обнаружено 16 ошибок. Время между обнаружением ошибок (Xi) приведено в таблице 3.3.

Таблица 3.3 – Время между ошибками (Xi) в часах:

|  |  |
| --- | --- |
| Номер ошибки (i) | Время между ошибками (Xi) |
| 1 | 5 |
| 2 | 8 |
| 3 | 10 |
| 4 | 12 |
| 5 | 14 |
| 6 | 7 |
| 7 | 9 |
| 8 | 16 |
| 9 | 17 |
| 10 | 18 |
| 11 | 19 |
| 12 | 20 |
| 13 | 21 |
| 14 | 22 |
| 15 | 23 |
| 16 | 25 |

Метод оценивает вероятность безотказной работы веб-приложения. Метод использует формулы 3.13 и 3.14 для вычисления этого значения:

|  |  |
| --- | --- |
|  | (3.13) |

|  |  |
| --- | --- |
|  | (3.14) |

Для расчёта коэффициента B необходимо найти решения уравнения 3.15:

|  |  |
| --- | --- |
|  | (3.15) |

где A – константа, зависящая от данных и равная значению, которое подходит под условие 3.16:

|  |  |
| --- | --- |
|  | (3.16) |

Для выполнения расчета примем A = 9.

Для нахождения B выполним замену (заменим B + 1 на m) и найдем минимально возможную разность между и . Также на m накладывается условие, что оно должно быть больше, чем n + 1. Такое число отражает число прогнозируемых (пока не обнаруженных) ошибок.

Функции f(m) и g(m,A) определяются следующим образом по формулам 3.17 и 3.18:

|  |  |
| --- | --- |
|  | (3.17) |

|  |  |
| --- | --- |
|  | (3.18) |

В ходе расчетов для различных значений m (диапазон значений был выбран от 17 до 80) получаем, что минимальная разность достигается при m=51, что соответствует общему числу ошибок B=50 (так как B=m−1).

Неизвестный коэффициент *K* рассчитывается по формуле 3.19:

|  |  |
| --- | --- |
|  | (3.19) |

Подставляя найденное значение в формулу, получаем:

|  |  |
| --- | --- |
|  | (3.20) |

Интенсивность отказов λ(t)после обнаружения 16 ошибок рассчитывается по формуле 3.13 и равно:

|  |  |
| --- | --- |
|  | (3.21) |

Вероятность безотказной работы *P*(*t*) на интервале t=1*t*=1 час определяется по формуле 3.14 и равно:

|  |  |
| --- | --- |
|  | (3.22) |

Среднее время до появления следующей ошибки (i + 1) рассчитывается по формуле 3.23:

|  |  |
| --- | --- |
|  | (3.23) |

Таким образом получаем, что время до появления следующей ошибки равно приблизительно 18 часов.

Время до окончания тестирования оценивается как сумма времени до обнаружения оставшихся ошибок (формула 3.24):

|  |  |
| --- | --- |
|  | (3.24) |

Предположив, что ошибки будут находиться через среднее значение времени, получим = 108 дней.

Таким образом, модель Джелинского – Моранды позволяет оценить надежность программного средства и спрогнозировать время, необходимое для обнаружения и исправления всех ошибок.

## 3.3 Расчет надежности по модели Муса

Проведем расчет надежности по модели Муса. В этой модели надежность программного средства на этапе эксплуатации оценивается по результатам тестирования. Пусть Т – суммарное время тестирования, М – число отказов, произошедших за время тестирования. Тогда по модели Муса средняя наработка до отказа после тестирования определяется по формуле (3.25).

|  |  |
| --- | --- |
|  | (3.25) |

В этой формуле τ0 – средняя наработка до отказа до начала тестирования, С – коэффициент, учитывающий уплотнение тестового времени по сравнению с временем реальной эксплуатации. Неизвестный параметр τ0 можно оценить из соотношения (3.26).

|  |  |
| --- | --- |
|  | (3.26) |

В этой формуле N – первоначальное число ошибок в программной средстве, которое можно оценить с помощью другой модели, позволяющей определить N на основе статистических данных, полученных при тестировании. К – коэффициент проявления ошибок, который определяется эмпирическим путем по однотипным программам, обычно это значение изменяется от 1,5·10–7 до 4·10–7. f – средняя скорость исполнения программы (A), деленная на число команд (операторов (B)).

Средняя скорость исполнения одного оператора программного средства определяется по формуле (3.27).

|  |  |
| --- | --- |
|  | (3.27) |

Длительности этапов тестирования составляют t1 = 50 ч, t2 = 42 ч, t3 = 36 ч. Число отказов на первом этапе m1 =8, на втором – m2 = 5, на третьем – m3 = 3. Средняя скорость исполнения программного средства = 107 операторов/ч, количество операторов в программном средстве = 1050. Определим надежность системы для периода эксплуатации t = 128 ч.

Найдем среднюю скорость выполнения одного оператора по формуле (3.28):

(3.28)

Первоначальное количество ошибок в программном средстве N равно 16. Коэффициент проявления ошибок K примем равным 2,75·10-7. Найдем формуле (3.29).

(3.29)

Примем значение коэффициента С = 5. Тогда средняя наработка до отказа после тестирования на этапе эксплуатации программного обеспечения рассчитывается по формуле (3.25).

Найдем надежность программного средства для периода эксплуатации t равному 128 часам по формуле (3.23).

|  |  |
| --- | --- |
|  | (3.23) |

Надежность равна:

Таким образов в этом разделе были посчитаны метрики надежности веб-приложения для усовершенствованной визуализации мониторинга качества воды с использованием цветных гексагонов на карте города Минска по следующим моделям Муса, Джелинского – Моранды и по модели сложности.

# 4 ТЕХНИКО-Экономическое обоснование разработки и использования Веб-приложение по усовершенствованию визуализации мониторинга качества воды посредством введения цветных гексагонов на карте города Минска

## 4.1 Характеристика разработанного веб-приложения по индивидуальному заказу

Данный раздел посвящен экономическому обоснованию дипломного проекта на тему «Веб-приложение для усовершенствованной визуализации мониторинга качества воды с использованием цветных гексагонов на карте города Минска». Проект относится к категории разработок по индивидуальному заказу и направлен на повышение эффективности анализа и представления данных о качестве воды в городской среде. Так как разработка программного обеспечения ведется для сторонней организации, для экономического обоснования выбрана методика обоснования разработки по индивидуальному заказу.

Основная цель разработки — улучшение визуализации данных мониторинга качества воды за счет интерактивной карты с цветовым кодированием в форме гексагонов. Это позволит выявлять проблемные зоны, упростит анализ экологической обстановки и повысит наглядность представления информации.

Текущий формат отображения качества воды состоит из карты, с отметкой местоположения в виде пина на карте, а также таблицы с различными характеристиками воды. Такой формат предполагает, что пользователь обладает достаточной информацией о предельно допустимых значениях, может сравнивать и оценивать риски тех или иных параметрах для здоровья. Разрабатываемое приложение позволит автоматизировать этот процесс для конечного пользователя в следствие использования цветовой разметки.

Область применения проекта — экология и охрана окружающей среды. Основными пользователями приложения будут граждане, интересующиеся текущей обстановкой качеств воды в Минске. Для пользователей приложение станет удобным инструментом для оценки качества воды в свой районе.

Основным конкурентным преимуществом приложения является цветовое отображение по областям города Минска с помощью гексагонов на карте. Помимо этого, приложение поддерживает историческую информацию о данных: можно отобразить на карте данные за конкретный период времени.

Экономическая оценка целесообразности инвестиций в разработку и использование программного средства осуществляется на основе расчета и оценки следующих показателей: чистый дисконтированный доход, рентабельность инвестиций и простой срок окупаемости инвестиций.

## 4.2 Расчет основных затрат на разработку

## 4.2.1 Расчет затрат на основную заработную плату разработчикам

Для расчета затрат на разработку программного средства в первую очередь необходимо рассчитать основную заработную плату команды разработчиков. Расчет осуществляется исходя из состава и численности команды, размера месячной заработной платы каждого участника команды, а также трудоемкости работ, выполняемых при разработке программного средства отдельными исполнителями по формуле:

(4.1)

где Кпр ‒ коэффициент премий (равный 1,5);

*n* ‒ категории исполнителей, занятых разработкой программного средства;

Зчi ‒ часовая заработная плата исполнителя *i*-й категории, р.;

*ti* ‒ трудоемкость работ, выполняемых исполнителем *i*-й категории, определяется исходя из сложности разработки программного обеспечения и объема выполняемых им функций, ч.

На 2025 год расчетная норма рабочего времени для пятидневной рабочей недели составляет 168 часов, 8 часов работы в день, среднемесячная расчетная норма рабочего времени – 21 день.

В разработке были задействованы следующие сотрудники: бизнес-аналитик, программист, тестировщик. Часовая заработная плата каждого сотрудника определена как результат деления месячной заработной платы (оклада) на количество рабочих часов в месяце. Расчет основных затрат представлен в таблице 4.1.

Таблица 4.1 ‒ Расчет затрат на основную заработную плату разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р | Часовой оклад, р | Трудоемкость работ, Р | Итого, р |
| Бизнес-аналитик | 1463 | 9 | 25 | 225 |
| Программист | 2091 | 12 | 450 | 5400 |
| Тестировщик | 1826 | 10 | 120 | 1200 |
| Итого | | | | 6825 |
| Премия и иные стимулирующие выплаты (50%) | | | | 3412 |
| Основная заработная плата разработчиков | | | | 10237 |

## 4.2.2 Расчет затрат на дополнительную заработную плату разработчикам

Дополнительная заработная плата ‒ это оплата за сверхурочный труд, различные трудовые успехи и надбавки за особые условия труда команды и включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах (составляет 20%) к основной заработной плате по следующей формуле:

(4.2)

где Зо ‒ затраты на основную заработную плату;

Нд ‒ норматив дополнительной заработной платы, 20%.

Подставим значение в формулу (4.2) и вычислим Зд:

Согласно расчетам, затраты на дополнительную заработную плату разработчикам составит 2047 рубля.

## 4.2.3 Расчет отчислений на социальные нужды

В расчете отчислений на социальные нужды учитываются обязательные платежи по установленным законодательством тарифам в фонд социальной защиты населения, а также затраты предприятия на обязательное социальное медицинское страхование для определенных категорий работников в соответствии с законодательством. Размер отчислений в фонд социальной защиты населения и на обязательное страхование определяется в соответствии с действующими законодательными актами Республики Беларусь и рассчитывается по установленной формуле:

(4.3)

где Нсоц ‒ норматив отчислений на социальные нужды, %.

Согласно законодательству Республики Беларусь, отчисления на социальные нужды составляют 29% в фонд социальной защиты и 6% на обязательное страхование. Подставим результаты вычислений в формулу (4.3) и вычислим Рсоц:

Согласно расчетам, размер отчислений в фонд социальной защиты и на обязательное страхование составляет 4299 рублей.

## 4.2.4 Расчет затрат на прочие расходы

Прочие расходы связаны с функционированием организации-разработчика в целом, например: затраты на аренду офисных помещений, отопление, освещение, амортизацию основных производственных фондов и так далее. При расчете данной статьи затрат учитывается норматив прочих затрат в целом по организации. В данном случае норматив прочих затрат равен 20 %. Размер затрат на прочие расходы рассчитывается по формуле:

(4.4)

где Ннз ‒ норматив прочих затрат в целом по организации, 30 %.

Подставим значение из выражения в формулу (4.4) и произведем расчет Рпр:

Согласно расчетам, размер затрат на прочие расходы составляет 3071 рубля.

## 4.2.5 Расчет суммы затрат на разработку

Общая сумма затрат на разработку рассчитывается путем суммирования основной заработной платы, дополнительной заработной платы, отчислений на социальные нужды, прочих затрат. Формула расчета имеет следующий вид:

(4.5)

Подставим результаты вычислений в формулу (4.5) и произведем расчет Зр:

Согласно расчетам, сумма затрат на разработку составляет 19654 рубля.

## 4.2.6 Расчет плановой прибыли, включаемой в цену программного средства

Плановая прибыль, включаемая в цену программного средства, рассчитывается по формуле:

(4.6)

В данном случае рентабельность затрат на разработку программного средства установили на уровне 25%. Подставим значение из выражения в формулу (4.6) и произведем расчет Пп.с.:

Исходя из расчетов, плановая прибыль, включаемая в цену программного средства, составляет рублей.

## 4.2.7 Расчет отпускной цены программного средства

Отпускная цена программного продукта представляет собой сумму затрат на заработную плату и плановой прибыли. Рассмотрим формулу расчета отпускной цены программного средства:

(4.7)

Подставим результат вычислений и произведем расчет Цп.с.:

Исходя из расчетов, отпускная цена программного средства составляет 24567,5 рублей.

## 4.2.8 Результаты расчета затрат на разработку и цены веб-приложения

В данном подразделе были рассчитаны необходимые статьи для расчета затрат на разработку и для расчета цены программного средства, а именно: основная заработная плата разработчиков, дополнительная заработная плата разработчиков, отчисления на социальные нужды, прочие расходы и плановая прибыль. Результаты расчетов представлены в таблице 4.2.

Таблица 4.2 ‒ Результаты расчета цены на разработку программного средства

|  |  |
| --- | --- |
| Наименование статьи затрат | Сумма, р. |
| 1 Основная заработная плата разработчиков | 10237 |
| 2 Дополнительная заработная плата разработчиков | 2047 |
| 3 Отчисления на социальные нужды | 4299 |
| 4 Прочие расходы | 3071 |
| 5 Всего затраты на разработку | 19654 |
| 6 Плановая прибыль | 4913,5 |
| 7 Цена программного средства | 24567,5 |

## 4.3 Расчет результата от разработки и реализации веб-приложения

Для организации-разработчика экономическим эффектом является прирост чистой прибыли, полученной от разработки и реализации программного средства заказчику. Так как программное средство будет реализовываться организацией-разработчиком по отпускной цене, сформированной на основе затрат на разработку, то экономический эффект, полученный организацией-разработчиком, в виде прироста чистой прибыли от его разработки, определяется по формуле:

(4.8)

где Пп.с. ‒ прибыль, включаемая в цену программного средства, р;

Нп   ‒ ставка налога на прибыль согласно действующему законодательству, (по состоянию на 01.01.2024 г. – 20%).

Подставим результат вычисления в формулу (4.8) и произведем расчет ΔПч:

Исходя из расчетов, экономический эффект составляет 3930,8 рублей.

Для организации-заказчика расчет экономического эффекта от использования программного обеспечения, разработанного по индивидуальному заказу сторонней организацией, осуществляется в соответствии с методикой расчета основных видов экономического эффекта.

Экономия на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ определяется по формуле:

(4.9)

где Кпр – коэффициент премий (по фактическим данным предприятия или в диапазоне 1,5-2);

*tр*без п.с, *t*рс п.с – трудоемкость выполнения работ сотрудниками до и после внедрения программного средства, ч;

*Т*ч – часовой оклад (часовая тарифная ставка) сотрудника, использующего программное средство, р;

*N*п – плановый объем работ, выполняемых сотрудником;

Нд – норматив дополнительной заработной платы;

Нсоц – ставка отчислений от заработной платы, включаемых в себестоимость (34,6%).

Подставим результат вычисления в формулу (4.9) и произведем расчет Эз.п:

Экономия на заработной плате и начислениях на заработную плату в результате сокращения численности работников определяется по формуле:

(4.10)

где *n* – категории работников, высвобождаемых в результате внедрения программного средства;

ΔЧ*i*  – численность работников *i*-й категории, высвобожденных после внедрения программного средства, чел.;

З*i*    – годовая заработная плата высвобожденных работников *i*-й категории после внедрения программного средства, р.;

Нсоц – норматив отчислений от заработной платы в соответствии с законодательством, %.

Экономия на заработной плате и начислениях на заработную плату в результате сокращения численности работников составляет 0 р., поскольку количество сотрудников не изменилось после внедрения веб-приложения.

Экономия на материальных ресурсах в результате внедрения программного средства определяется по формуле:

(4.11)

где Кт.р. – коэффициент транспортных расходов (по данным предприятия или 1,05-1,2);

Нрбез п.с, Нрсп.с – норма расхода материальных ресурсов при выполнении работ сотрудниками до и после внедрения программного средства, нат. ед.;

Цм – цена за единицу материального ресурса, р.;

*N*п – плановый объем работ, выполняемых сотрудником с использованием программного средства.

Экономия на материальных ресурсах так же равно 0 р., поскольку расход материальных ресурсов не изменился.

Экономическим эффектом при использовании программного средства является прирост чистой прибыли, полученной за счет экономии на текущих затратах предприятия, который рассчитывается по формуле:

(4.12)

где Этек – экономия на текущих затратах при использовании программного средства, р.;

ΔЗтекп.с– прирост текущих затрат, связанных с использованием программного средства, р;

Нп    – ставка налога на прибыль согласно действующему законодательству.

Таким образом экономический эффект при использовании программного средства составит

## 4.4 Расчет показателей экономической эффективности разработки и использования веб-приложения

Для организации-разработчика программного средства оценка экономической эффективности разработки осуществляется с помощью расчета рентабельности затрат на разработку программного средства. Рентабельность является одним из основных показателей эффективности предприятия с точки зрения использования привлеченных средств. Она представляет собой отношение суммы чистой приведенной прибыли, полученной за весь расчетный период, к суммарным приведенным затратам за этот же период и определяется по формуле:

(4.13)

где ΔПч ‒ прирост чистой прибыли, полученной от разработки программного средства организацией-разработчиком по индивидуальному заказу, р;

Зр ‒ затраты на разработку программного средства организацией-разработчиком, р.

Подставим результат вычисления в формулу (4.9) и произведем расчет Рз:

Рассчитанный показатель отображает, сколько чистой прибыли компания-разработчик получит от вложенных денег в разработку программного средства.

Так как сумма инвестиций больше суммы годового прироста чистой прибыли, для организации-заказчика рассчитывается несколько показателей экономической эффективности.

Для приведения доходов и затрат к настоящему моменту времени определяется коэффициент дисконтирования по формуле:

(4.14)

где *d* –   требуемая норма дисконта, которая по своему смыслу соответствует устанавливаемому инвестором желаемому уровню рентабельности инвестиций, доли единицы;

*t* – порядковый номер года, доходы и затраты которого приводятся к расчетному году;

*tp* – расчетный год, к которому приводятся доходы и инвестиционные затраты.

Норму дисконта принимаем равным ставке рефинансирования Национального банка Республики Беларусь – 9,5%. Расчетный период составит четыре года.

Таким образом, коэффициенты дисконтирования за каждый год составляют:

В течение первого года осуществляется разработка приложения, поэтому в первый год экономический эффект будет меньше планируемого. Для того, чтобы учесть этот факт, необходимо выяснить, сколько времени будет затрачено на разработку приложения.

Так как работа команды разработчиков осуществляется поэтапно, то затраченное время будет равно сумме трудоемкости работ команды, и составит 595 часов.

Таблица 4.3 – Расчет эффективности инвестиций (затрат) в реализацию проектного решения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Показатель | Значение расчетного периода по годам | | | |
| 1-й год | 2-й год | 3-й год | 4-й год |
| 1 Прирост чистой прибыли, р. | 12997,7 |  |  |  |
|  |  |  |  |  |
| 2 Дисконтированный резуль­тат, р | 12997,7 | 14602,4 | 13318,7 | 12195,4 |
| 3 Инвестиции в разработку, р. | 19654 | 0 | 0 | 0 |
| 4 Дисконтированные инвестиции, р. | 19654 | 0 | 0 | 0 |
| 5 Чистый дисконтированный доход по годам, р. | -6656,3 | 14602,4 | 13318,7 | 12195,4 |
| 6 Чистый дисконтированный доход нарастающим итого, р. | -6656,3 | 7946,1 | 21264,8 | 33460,2 |
| 7 Коэффициент дисконтирования, доли единицы | 1,00 | 0,91 | 0,83 | 0,76 |

В данном случае дисконтированный эффект нарастающим итогом превысит дисконтированные инвестиции на второй год. Простой срок окупаемости рассчитывается по формуле:

(4.15)

Таким образом, простой срок окупаемости инвестиций равен:

Чистый дисконтированный доход рассчитывается по формуле:

(4.16)

где – коэффициент дисконтирования, рассчитанный для года t.

Таким образом, чистый дисконтированный доход равен:

В результате проведения расчетов была определена необходимость разработки программного обеспечения, а также получен экономический эффект от использования данного программного продукта. По результатам проведенного экономического обоснования были получены следующие результаты:

1 Стоимость заказа на разработку программного средства автоматизации бэк-офисных процессов организации составила 24567,5 рублей.

2 Прирост чистой прибыли составил 16046,6 рублей.

3 Данная разработка имеет положительный экономический эффект в размере 81%.

4 По результатам проведенных расчетов, вложенные инвестиции должны окупится за 1,28 года, чистую прибыль стоит ожидать на второй год реализации проекта.

Таким образом, разработка и реализация по индивидуальному заказу веб-приложения для усовершенствованной визуализации мониторинга качества воды с использованием цветных гексагонов на карте города Минска с экономической точки зрения целесообразна.

# 5 ОХРАНА ТРУДА. МЕРОПРИЯТИЯ ПО ОБЕСПЕЧЕНИЮ ВЫСОКОЙ РАБОТОСПОСОБНОСТИ И СОЗДАНИЮ КОМФОРТНЫХ УСЛОВИЙ ТРУДА ПРИ РАЗРАБОТКЕ И ЭКСПЛУАТАЦИИ ВЕБ-ПРИЛОЖЕНИЯ

Веб-приложение по усовершенствованию визуализации мониторинга качества воды посредством введения цветных гексагонов на карте города Минска и его эргономическое обеспечение. Работа операторов, занимающихся разработкой веб-приложений, требует непрерывного взаимодействия с различными устройствами, такими как мониторы, ноутбуки, планшеты и другие средства отображения информации. Однако, при такой работе оператор может столкнуться с проблемами зрительного восприятия информации и формирования утомления зрительного анализатора, что в свою очередь может привести к ухудшению эффективности труда и появлению заболеваний глаз. Зрительное восприятие информации является сложным процессом, включающим в себя множество факторов, таких как размер и форма объекта, контрастность, яркость и цветность изображения, частота мерцания экрана. В процессе работы с СОИ оператор непрерывно воспринимает информацию, что может привести к утомлению зрительного анализатора и снижению эффективности работы [24]. Утомление зрительного анализатора проявляется в виде ухудшения качества восприятия информации, затруднения в концентрации внимания, увеличения количества ошибок при выполнении задач. Чтобы избежать таких последствий, необходимо учитывать особенности зрительного восприятия при разработке и уделять внимание организации рабочего пространства [25]. Среди главных причин утомления зрительного анализатора выделяют: низкое качество изображения на мониторе, плохая контрастность, размытость, искажения цветопередачи, неравномерная яркость. Все это приводит к тому, что глаза оператора вынуждены работать с большей интенсивностью, чтобы воспринять информацию.

Недостаточная или избыточная яркость освещения рабочего пространства. Отсутствие достаточного освещения или наличие слишком яркого света может привести к резкому контрасту между светом и монитором, что ухудшит качество восприятия информации и приведет к быстрому утомлению глаз.

Отсутствие возможности регулирования яркости и контрастности монитора. В процессе разработки веб-приложения некоторые изображения, отображаемые на экране, могут быть слишком яркими или слишком темными, что также может ухудшить качество восприятия информации и привести к утомлению глаз.

Неправильное расположение экрана и рабочего места. Если экран находится на неправильной высоте или на неправильном расстоянии от глаз оператора, это может привести к увеличению нагрузки на мышцы глаз и шеи, что в свою очередь может привести к болезненным ощущениям и утомлению.

Монотонность работы. Длительное время, проведенное за монитором, без перерывов и смены задач, может привести к утомлению зрительного анализатора и снижению эффективности работы оператора [26].

Одним из способов решения проблем, вызывающих проблемы со здоровьем у операторов, является соблюдение инженерно-психологических требований − специальных правил и рекомендаций, разработанных с учетом особенностей человеческого организма. Они помогают создать наиболее оптимальные условия для работы с информацией, минимизируя возможные проблемы со здоровьем. Такие требования могут предъявляться к различным системам и устройствам, используемым оператором, таким как средства отображения информации, устройства управления, средства связи [27]. Чтобы избежать проблем со зрительным восприятием и утомления глаз, необходимо учитывать следующие инженерно-психологические требования к средствам отображения информации и их расположению в рабочем пространстве.

Разрешение экрана. Высокое разрешение экрана позволяет получать четкие и детализированные изображения, что снижает утомляемость глаз. Оптимальным разрешением экрана считается 1920x1080 пикселей.

Мерцание экрана. Операторам рекомендуется выбирать мониторы с минимальным уровнем мерцания экрана. Оптимальный вариант - мониторы с частотой обновления экрана не менее 60 Гц.

Подсветка экрана. Экран должен быть равномерно подсвечен, без ярких и темных пятен. Также следует учитывать уровень яркости экрана и его контрастность. Зачастую производители мониторов предоставляют специальные шаблоны настройки яркости, контрастности и цвета для работы в разных условиях, таких как дневное светлое время, темное время суток или различные условия освещения, однако такие настройки не всегда могут соответствовать требованиям конкретного оператора. Таким образом, 82 оптимальный уровень яркости экрана зависит от типа работы и задач, выполняемых пользователем, а также от условий окружающей среды и настраивается индивидуально для каждого оператора.

Размер шрифта. Размер шрифта должен быть достаточно большим для легкого чтения, но не слишком крупным, чтобы не загружать глаза. Рекомендуемый размер шрифта − от 10 до 14 пунктов. Однако следует учитывать, что оптимальный размер шрифта может изменяться от конкретной отображаемой информации и особенностей зрительного восприятия оператора.

Расположение экрана. Экран должен быть расположен на уровне глаз, чтобы оператор мог смотреть на него без необходимости наклонять или поворачивать голову. Наклон экрана должен быть примерно 15 градусов, чтобы уменьшить напряжение в шее и спине. Расстояние от глаз до экрана также имеет значение. Если экран расположен слишком близко, то глаза будут напряжены, что может привести к усталости глаз и боли в голове. Если же экран расположен слишком далеко, то глаза будут напряжены, чтобы читать текст на экране, что также может привести к усталости глаз и боли в голове. Оптимальное расстояние от экрана до глаз составляет от 50 до 70 см [28].

Качество монитора. Мониторы с качественной матрицей, такой как IPS (In-Plane Switching) или PLS (Plane-to-Line Switching), обеспечивают более четкое и точное отображение изображений и цветов. Они также имеют более широкий угол обзора, что позволяет смотреть на экран с разных углов без потери качества изображения. IPS и PLS матрицы также обеспечивают более равномерную яркость на всей поверхности экрана и не имеют мерцания, что помогает снизить утомляемость глаз [29].

При разработке веб-приложения важно также учесть безопасность и удобство взаимодействия пользователей с приложением. Веб-приложение может использоваться в различных условиях и на различных устройствах. Поэтому разработчикам важно предусмотреть возможность настройки интерфейса приложения, чтобы пользователи могли изменить размеры и шрифты в соответствии с их индивидуальными потребностями. Важно также учитывать возможность адаптации веб-приложения для пользователей с ограниченными возможностями зрения, таких как люди с дальнозоркостью, близорукостью, цветовой слепотой. Например, для людей с цветовой слепотой необходимо предусмотреть возможность отображения информации в режиме повышенной контрастности.

Ключевую роль в зрительном восприятии информации играет освещение рабочего пространства. Освещение должно соответствовать определенным требованиям.

Интенсивность освещения. Оптимальный уровень освещенности рабочей поверхности должен быть не менее 500 лк, однако, в зависимости от характера работы, этот уровень может быть увеличен до 1000 лк или более. Равномерность освещения. Освещение должно быть равномерно по всей рабочей поверхности. При наличии ярких и темных пятен на экране или рабочей поверхности, глаза оператора быстро устают.

Тип и цвет света. Рекомендуется использовать лампы с природным белым светом (4000-4500 К) или дневным светом (5000-5500 К). Не рекомендуется применение ламп с желтым светом, которые могут вызывать усталость глаз.

Блики и отражения. Необходимо избегать бликов на экране и отражений от других объектов, таких как стены, окна или предметы на рабочей поверхности. Это можно достичь за счет расположения рабочего места под правильным углом и использования плотных штор или жалюзи для блокировки прямых лучей солнца.

Дополнительное освещение. При необходимости можно использовать дополнительное освещение, такое как настольные лампы или светодиодные ленты, чтобы улучшить видимость на определенных участках рабочей поверхности [30].

Еще одним важным фактором, влияющим на здоровье и утомляемость зрительного анализатора, является режим труда и отдыха оператора. Основные рекомендации в этой области включают разнообразие задач. Операторам следует предоставлять разнообразные задачи для выполнения, чтобы избежать монотонной работы и уменьшить утомляемость. Регулярные перерывы помогают снизить усталость и напряжение глаз, а также уменьшить риск возникновения мышечной травмы. Рекомендуется делать перерывы каждый час на 10-15 минут.

Физические упражнения, упражнения для глаз помогают уменьшить усталость и напряжение в мышцах.

Регулярные проверки зрения. Операторы должны регулярно проверять зрение, чтобы выявить возможные проблемы и предотвратить их развитие.

Правильная организация рабочего места. Рабочее место должно быть организовано таким образом, чтобы минимизировать напряжение и усталость оператора. Это может включать в себя правильное расположение средств отображения информации, использование эргономичных кресел и столов, а также правильное расположение клавиатуры и мыши.

Регулярное обслуживание оборудования помогает предотвратить неожиданные поломки и снизить риск травмы, связанной с работой с оборудованием, которое неисправно или не настроено правильно [31].

В рамках разработки дипломного проекта учтены все вышеперечисленные рекомендации. Был использован монитор с частотой 60 Гц и разрешением экрана 1920х1080 пикселей, расположенный на уровне глаз на расстоянии около 50 см. Яркость монитора настроена на оптимальный уровень, обеспечивающий четкое восприятие изображения. Размер шрифта варьировался в зависимости от текущих задач, однако всегда был достаточно крупным для комфортного восприятия информации без излишнего напряжения глаз. Разработка велась в дневное время при естественном дневном освещении. Для минимизации излишней яркости и отражений использованы жалюзи. В ходе разработки регулярно осуществлялись перерывы и смена задач для предотвращения монотонности труда. Таким образом, оптимизация зрительного взаимодействия оператора со средствами отображения информации при разработке веб-приложения является важным аспектом охраны труда. Инженерно-психологические требования к средствам отображения информации и их расположению в рабочем пространстве, требования к организации, качественным и количественным характеристикам освещения рабочего места оператора, а также оптимизация режима труда и отдыха оператора − все эти факторы влияют на здоровье и безопасность работы оператора при разработке веб-приложения.

# ЗАКЛЮЧЕНИЕ

Разработанное веб-приложение представляет собой современное решение для визуализации данных о качестве воды в Минске. Оно успешно преодолевает ограничения традиционных табличных форматов, предлагая интерактивную гексагональную карту с цветовым кодированием параметров. Это позволяет пользователям оперативно оценивать ситуацию в разных районах города без необходимости анализа сложных технических данных.

Основу системы составляет трехзвенная архитектура, сочетающая удобный интерфейс, мощную серверную часть и надежное хранилище данных. Особое внимание было уделено алгоритмам обработки информации, которые обеспечивают точность расчетов и корректность визуализации. Технологический стек, включающий Django, React и PostgreSQL, позволил создать масштабируемое решение, готовое к увеличению нагрузки и расширению функциональности.

Экономическое обоснование проекта подтвердило его практическую значимость и финансовую устойчивость. Расчеты показали, что система не только окупится в кратчайшие сроки, но и сможет приносить стабильную прибыль при дальнейшей эксплуатации. Особую ценность представляет социальный эффект - повышение экологической грамотности населения и создание прозрачной системы мониторинга.

Важным достижением проекта стала разработка уникального алгоритма преобразования данных о качестве воды в цветовую схему гексагонов. Этот подход обеспечивает мгновенное визуальное восприятие информации, что особенно важно для оперативного принятия решений. Система успешно прошла тестирование на реальных данных, подтвердив свою надежность и точность отображения.

Перспективы развития связаны с интеграцией дополнительных источников данных и внедрением персонализированных сервисов для пользователей. Система обладает значительным потенциалом для адаптации к другим городам и экологическим параметрам. Реализованный проект служит наглядным примером того, как современные технологии могут способствовать решению актуальных социально-экологических задач, обеспечивая при этом экономическую эффективность и удобство использования.