МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Мега факультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Лабораторная работа № 4**

**Анализ и моделирование программной архитектуры и архитектуры данных программного средства (средств) автоматизации ИТ-процессов во взаимосвязи с функциональной и информационной архитектурой ИС.**

Выполнили студенты группы M33091

Джахан Исрат,

Кирьянов Глеб Дмитриевич,

Заднепровский Николай Владимирович,

Мальков Александр Ахмедович

Санкт-Петербург

2024

**Цель работы:**

Построить модели и провести сопоставление функциональной, информационной, программной архитектур и архитектуры данных отдельного программного средства. В качестве программного средства рекомендуется выбрать небольшое opensource решение, хотя можно использовать и проприетарное ПО, если есть доступ к кодовой базе и возможность представления его архитектуры. Если анализируемый продукт имеет достаточно богатый функционал, допустимо (по согласованию с преподавателем), раскрыть в работе только часть функций и их программной реализации.

**Порядок выполнения работы:**

1. Описать типовые ИТ-процессы (сбора, обработки, хранения, передачи и предоставления информации) для автоматизации которых предназначено определенное в теме средство автоматизации, конкретных информационных объектов, формируемых, хранимых, обрабатываемых или передаваемых этим средством, целей и показателей качества этих ИТ-процессов. Выбрать методологию и в соответствии с ее правилами сформировать набор диаграмм, дающих формальное описание процессов. Сделать выводы о функциональных требованиях к средствам автоматизации со стороны смоделированных процессов. При наличии возможности описать и обосновать нефункциональные требования к средству автоматизации.
2. Описать программную архитектуру анализируемого программного средства на уровне выделения отдельных программных компонентов, библиотек, модулей, описания основных классов, логики их взаимодействия, а также описать архитектуру данных, включая описание используемых стандартных типов данных, сложных пользовательских типов данных, организации хранения структурированных данных в хранилищах. Для каждого программного компонента привести описание его назначения, а также технологий, используемых при его разработке. Все описания обязательно дополняются визуальными моделями, построенными в соответствии с требованиями нотации UML.
3. Сопоставить функциональную, информационную, программную архитектуры и архитектуру данных для выбранного программного средства автоматизации ИТ- процессов. В результате должны быть построены как минимум два обязательных сопоставления. Первое – сопоставление функциональной и программной архитектуры. Должно быть указано, какие функции, относящиеся к каким функциональным компонентам реализованы с помощью каких программных компонентов, а также какие программные компоненты используются одновременно для реализации различных функций. Второе – сопоставление информационной архитектуры и архитектуры данных. Должно быть указано, какие информационные объекты реализованы с помощью каких структур данных, как организовано хранение информационных объектов. Отдельно рекомендуется описать, как обеспечивается целостность данных при работе с информационными объектами, транзакционность выполнения операций,

непротиворечивость и уникальность идентификаторов и другие технические аспекты реализации информационных объектов на уровне работы с данными.

**Описание проекта:**

Проект "WaterGuru": инновационное решение для контроля качества воды в бассейнах

WaterGuru представляет собой интеллектуальную систему мониторинга качества воды в бассейнах, включающую мобильное приложение для смартфонов и опциональный беспроводной прибор. Разработанное на платформе Flutter, приложение совместимо с iOS и Android и обеспечивает полноценное взаимодействие пользователя с информацией о состоянии воды в бассейне.

Основой проекта является устройство WaterGuru Sense, которое, будучи установленным в бассейне, обеспечивает сбор данных о состоянии воды и их передачу через Wi-Fi на облачный сервер. Пользователи приложения получают доступ к актуальным данным об уровне pH, температуре воды, содержании хлора и других параметрах, с возможностью автоматического ежедневного мониторинга.

При отсутствии устройства WaterGuru Sense пользователи могут вручную вводить показатели воды, полученные собственными измерениями, и получать персональные рекомендации по поддержанию оптимального состояния воды в бассейне с использованием различных химических реагентов.

Приложение WaterGuru также предлагает удобный инструментарий для анализа истории изменения качества воды, предоставляя графики и статистику, которые помогают владельцам бассейнов оптимизировать процесс ухода за водой и сохранять идеальное состояние воды для купания.

Центральным элементом системы является облачный сервер, на котором развернута база данных postgreSQL 15 для хранения и обработки всех собранных данных. Облачное хранилище обеспечивает высокую доступность и надежность информации, а также предоставляет возможность для дальнейшего развития и масштабирования проекта.

**Часть 1.  
Типовые ИТ-процессы проекта WaterGuru включают следующие основные этапы:**

1. Сбор данных о качестве воды (автоматический или ручной).  
Автоматический:  
1. Запросили измерение из приложения(либо автоматически отправили запрос по расписанию)  
2. Запрос дошёл до клауда, клауд обработал запрос и отправил запрос на измерение в прибор(девайс WaterGuru Sense)  
3. WaterGuru Sense начал измерение, которое занимаем 15 минут  
4. WaterGuru Sense провёл измерения и отправил данные SensorData в облако

2. Обработка данных и формирование рекомендаций по уходу за водой в бассейне.  
1. Запрос дошёл до клауда и клауд начал обрабатывать значения, отправленные датчиком.  
2. Результатом обработки WaterQualityData класс, который содержит в себе  
 \*значение pH  
 \*значение Free Chlorine  
 \*Температуру воды  
 \*Дату измерения  
3. Вместе с рассчитанными значениями для значений, перечисленных выше, также рассчитались рекомендации, для улучшения общего состояния воды в бассейне.

3. Хранение данных и обеспечение их доступности для пользователей.  
1. Данные, рассчитанные на предыдущем шагу сохранились в базу данных postrgeSQL  
2. Данные об измерении сохраняются по ключам, ассоциированным с бассейном.  
3. Все бассейны, относящиеся к одному пользователю, ассоциируются с ним по ключу аккаунта.

4. Передача данных между устройствами (прибором, смартфоном и облачным сервером).  
1. После того, как устройство произвело измерение, данные отправляются на облако по HTTPS протоколу.  
2. Облако рассчитывает на основе полученных данных состояние воды в бассейне и отправляет эти данные на мобильное приложение для отображения также по HTTPS протоколу.

5. Предоставление информации и статистики пользователю через мобильное приложение.  
1. Приложения, получив новый комплект данных о состоянии воды в бассейне отображает их, предоставляя числовое значение, статус(низкий уровень, высокий, нормальный и т.д.) и дату измерения  
2. Также в приложении есть доступ к истории конкретного показателя, которая отображает измеренные данные за последний день, неделю, месяц и год в виде графиков со значениями.

**Ключевые информационные объекты, формируемые и обрабатываемые в рамках проекта WaterGuru, включают:**

* Данные о качестве воды (уровень pH, температура, содержание хлора и др.).
* Рекомендации по поддержанию качества воды (добавление химикатов, добавление свежей воды, и др.).
* История изменений качества воды и статистика.

**На основе анализа типовых ИТ-процессов можно сформировать следующие функциональные требования к средствам автоматизации проекта WaterGuru:**

1. Сбор и передача данных с датчика WaterGuru Sense на облачный сервер.

2. Расчет рекомендаций на основе введенных данных о качестве воды (для пользователей без датчика).

3. Хранение и обработка данных на облачном сервере с использованием базы данных postgreSQL 15.

4. Визуализация данных и статистики в мобильном приложении (графики, таблицы, уведомления).

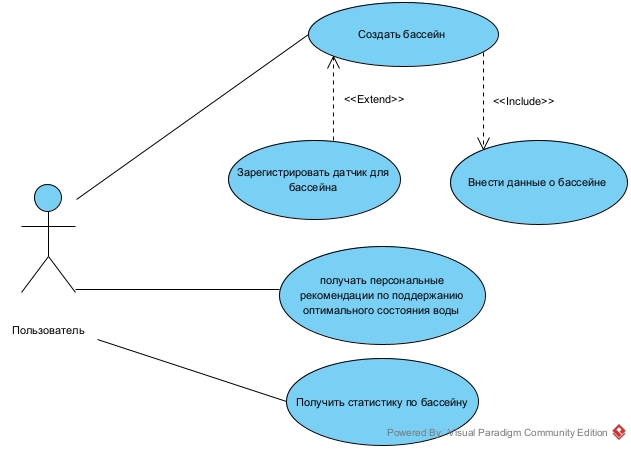
5. Обеспечение доступности и синхронизации данных между устройствами и облачным сервером.

**Потенциальные нефункциональные требования к средствам автоматизации могут включать:**

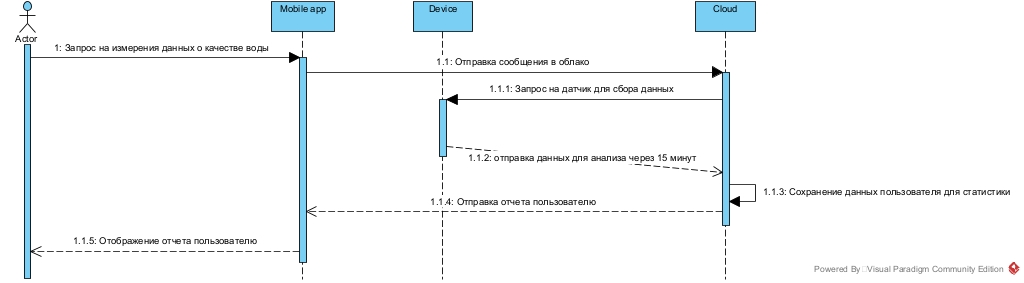
1. Быстродействие и отзывчивость мобильного приложения и облачного сервера.
2. Надежность и безопасность хранения и передачи данных.
3. Простота и интуитивность интерфейса мобильного приложения.
4. Совместимость с различными платформами (iOS и Android) и разнообразными устройствами (смартфоны, планшеты).
5. Масштабируемость архитектуры для поддержки роста числа пользователей и увеличения объема обрабатываемых данных.
6. Работа приложения и датчика в условиях ограниченного доступа к интернету или временного отсутствия связи.

**С учетом описанных процессов и требований можно разработать набор диаграмм, отражающих ключевые этапы и операции в рамках проекта WaterGuru. Среди них могут быть следующие диаграммы:**

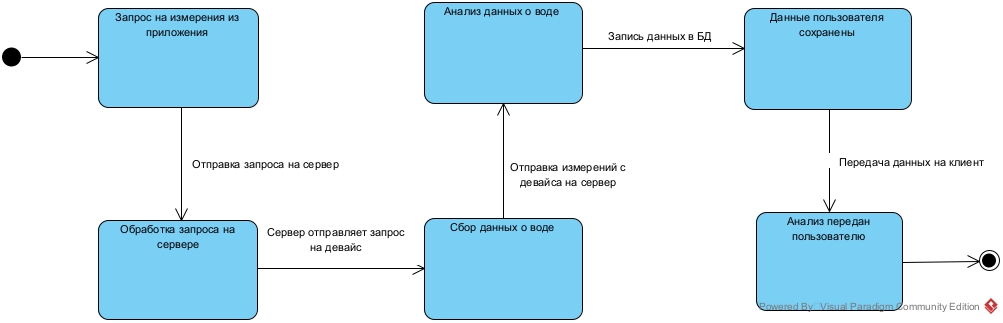
1. Диаграмма процессов, отражающая взаимодействие акторов (пользователь, датчик, мобильное приложение, облачный сервер) и основные этапы работы системы.



2. Диаграмма последовательности, показывающая порядок операций и передачи данных между акторами.

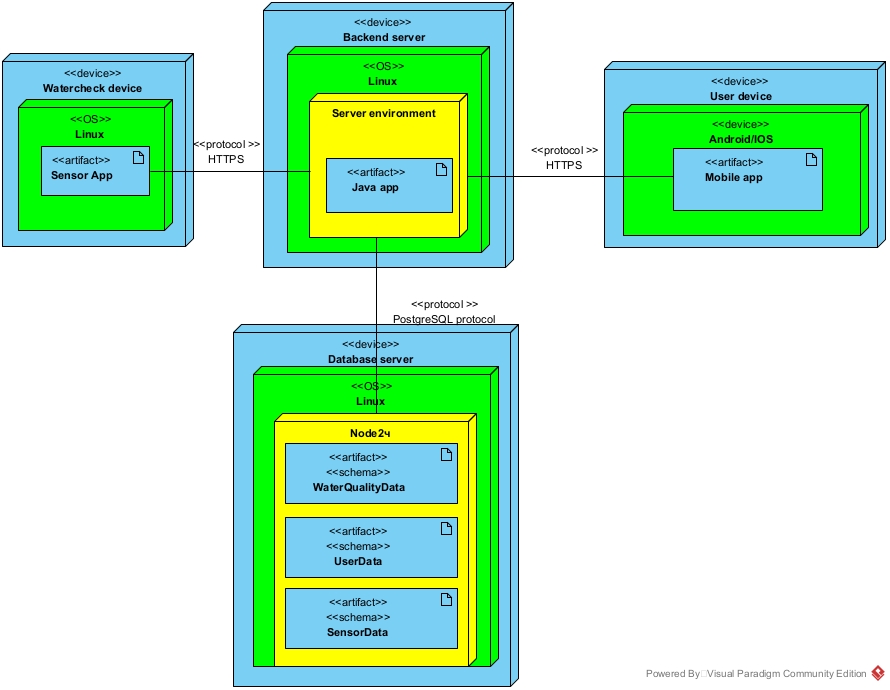


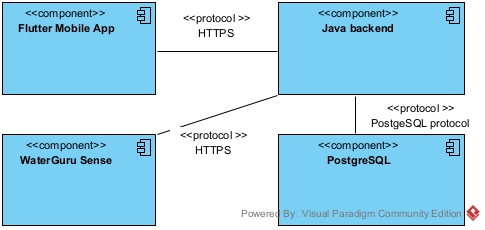
3. Диаграмма состояний, иллюстрирующая возможные состояния информационных объектов и переходы между ними.



На основе смоделированных процессов можно сформулировать детализированные функциональные и нефункциональные требования к средствам автоматизации проекта WaterGuru, что будет способствовать успешной реализации и оптимизации проекта с точки зрения удовлетворения потребностей пользователей и обеспечения высокого качества предоставляемых услуг.

Набор диаграмм, дающих формальное описание процессов. Модели должны демонстрировать анализируемые процессы с точностью до отдельных операций, позволять для этих операций определить акторов и информационные объекты,использующиеся в них.





**Часть 2.**

**Программная архитектура проекта WaterGuru состоит из следующих основных компонентов:**

1. Мобильное приложение WaterGuru:

* Разработано на платформе Flutter для совместимости с iOS и Android.
* Содержит модули для взаимодействия с облачным сервером, обработки данных, отображения информации и управления уведомлениями.
* Основные классы: WaterQualityData, UserData, NotificationManager, AdviceData.

2. WaterGuru Sense:

* Беспроводной датчик для сбора информации о качестве воды.
* Интегрируется с Wi-Fi для передачи данных на облачный сервер.
* Основные классы: SensorController, SensorData, WiFiManager, MeasurementData.

3. Облачный сервер:

* Основан на базе данных postgreSQL 15 для хранения и обработки данных.
* API для взаимодействия с мобильным приложением и WaterGuru Sense.
* Основные классы: DataController, APIHandler, DatabaseManager.

4. Библиотеки и технологии:

* Flutter (разработка мобильного приложения).
* Wi-Fi (подключение датчика к интернету).
* postgreSQL (хранение и обработка данных на сервере).
* REST API (взаимодействие между мобильным приложением, датчиком и сервером).

**Архитектура данных включает следующие структуры:**

1. Стандартные типы данных: строки, числа с плавающей точкой, целые числа, булевы значения и даты.

2. Сложные пользовательские типы данных:

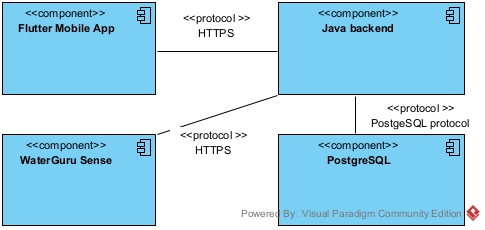
* WaterQualityData: класс, описывающий данные о качестве воды (pH - float, температура - float, хлор - float).
* UserData: класс, содержащий информацию о пользователе и настройки уведомлений. Имя - string. Фамилия - string. Номер телефона - int. Пароль - string. Настройки уведомлений - boolean (вкл выкл)
* SensorData: класс, описывающий данные, полученные от WaterGuru Sensor.  
   Время измерения - DateTime, уровень заряда батарейки - int, количество оставшийся кассеты - DateTime (эстимэйт когда она кончится))  
  набор rgb значений, не вдаваясь в подробности комплект вида   
  (  
  {[0..255],[0..255],[0..255]}  
  {[0..255],[0..255],[0..255]}  
  {[0..255],[0..255],[0..255]}  
  )

3. Организация хранения структурированных данных в хранилищах:

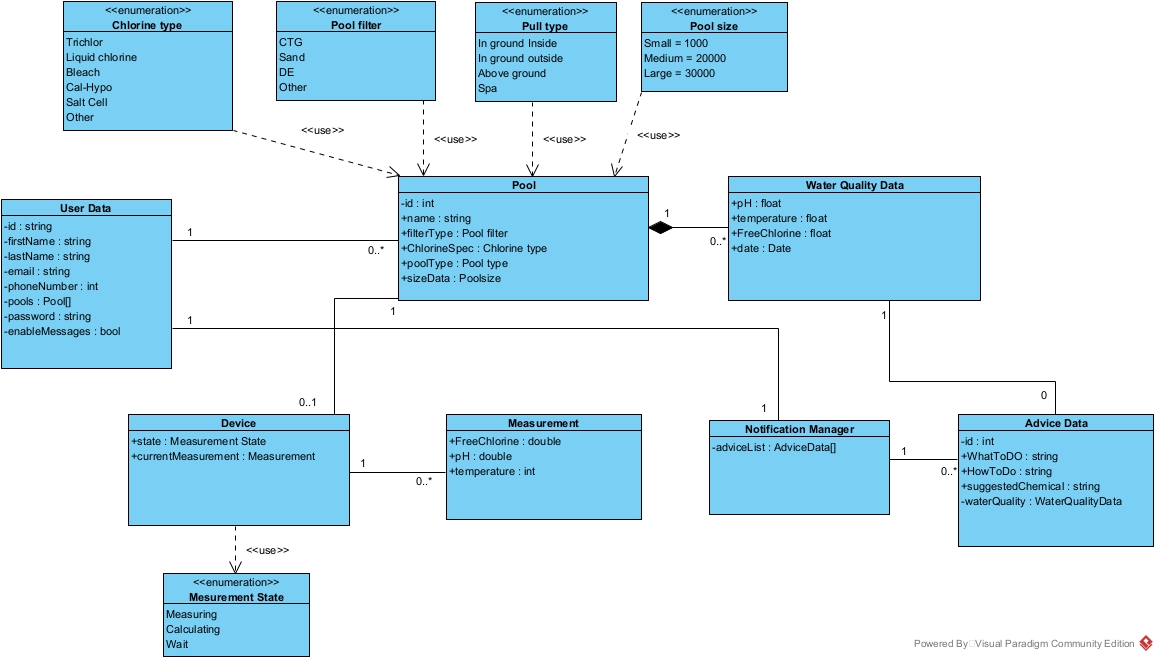
* В базе данных postgreSQL создаются таблицы для хранения информации о пользователях, датчиках и данных о качестве воды.
* Обеспечивается целостность данных и связи между таблицами с использованием внешних ключей и индексов.

**Для визуализации программной архитектуры и архитектуры данных можно использовать следующие диаграммы UML:**

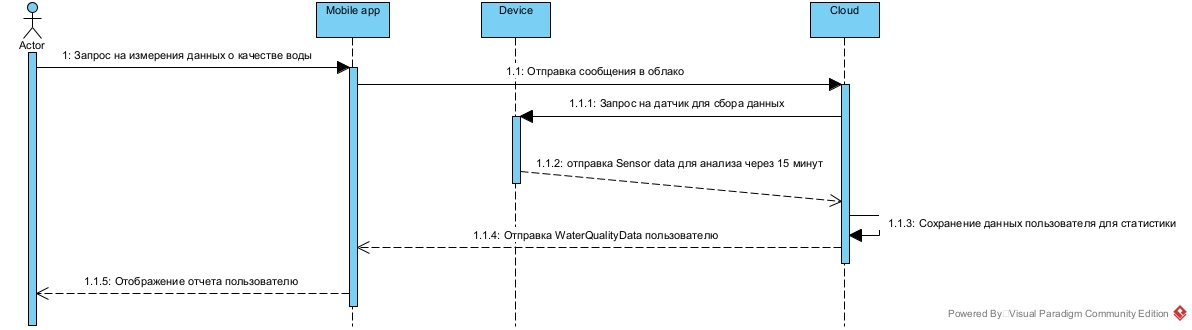
1. Диаграмма компонентов: показывает отношения между основными компонентами системы (мобильное приложение, WaterGuru Sensor и облачный сервер) и используемыми технологиями (Flutter, Wi-Fi, postgreSQL, REST API).



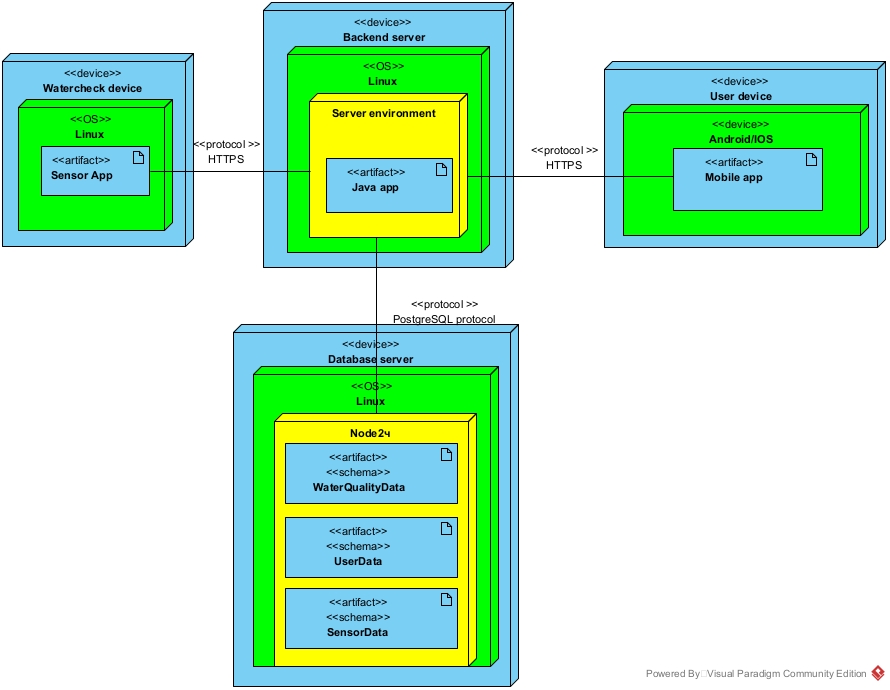
2. Диаграмма классов: иллюстрирует основные классы системы (WaterQualityData, UserData, SensorData, и другие), их атрибуты, методы и взаимосвязи между ними.



3. Диаграмма последовательности: отображает взаимодействие между классами и компонентами при выполнении ключевых операций, таких как сбор данных датчиком, передача данных на сервер и получение информации о качестве воды пользователем через мобильное приложение.



4. Диаграмма развертывания: представляет архитектуру системы на уровне оборудования, показывая размещение мобильного приложения на смартфонах, датчика WaterGuru Sensor в бассейне и облачного сервера с базой данных postgreSQL.



**На основе анализа программной архитектуры и моделей UML, можно сделать выводы о функциональных требованиях к средствам автоматизации:**

1. Мобильное приложение должно обеспечивать удобный интерфейс для взаимодействия пользователя с данными о качестве воды, уведомлениями и рекомендациями по уходу за водой.

2. WaterGuru Sensor должен автоматически собирать данные о состоянии воды и передавать их на сервер через Wi-Fi.

3. Облачный сервер должен обеспечивать надежное и быстрое хранение, обработку и передачу данных между датчиком и мобильным приложением.

4. Все компоненты системы должны быть легко масштабируемыми и готовыми к интеграции с дополнительными устройствами, сервисами и технологиями.

**Возможные нефункциональные требования к средству автоматизации включают:**

1. Производительность: быстрый отклик мобильного приложения, минимальные задержки при передаче данных между компонентами.

2. Надежность: обеспечение стабильной работы системы, устойчивости к сбоям и ошибкам.

3. Безопасность: защита данных пользователя, шифрование при передаче данных между компонентами и на сервере.

4. Удобство использования: интуитивно понятный интерфейс мобильного приложения, простота настройки и интеграции датчика WaterGuru Sensor, доступность подробных инструкций и поддержки для пользователей.

5. Совместимость: поддержка мобильного приложения на различных платформах (iOS и Android) и устройствах, а также возможность интеграции с различными моделями бассейнов и системами обслуживания бассейнов.

6. Масштабируемость: возможность расширения системы для обслуживания большого количества пользователей, бассейнов и дополнительных функций.

7. Экологичность и экономичность: использование энергоэффективных компонентов и оптимизация процессов обработки данных для снижения энергопотребления и затрат на обслуживание системы.

Чтобы удовлетворить все вышеуказанные требования, разработчики проекта WaterGuru должны были уделить особое внимание выбору подходящих технологий, оптимизации процессов обработки и передачи данных, а также созданию удобного и безопасного пользовательского интерфейса.

**Реализация проекта WaterGuru предполагала соблюдение следующих этапов разработки:**

1. Сбор и анализ требований: определение функциональных и нефункциональных требований, а также ожиданий пользователей от системы.

2. Проектирование архитектуры: разработка диаграмм UML, определение основных компонентов, классов, их взаимодействия и структуры данных.

3. Разработка и тестирование отдельных компонентов: создание мобильного приложения, датчика WaterGuru Sense, облачного сервера и базы данных, а также проверка их работоспособности и соответствия требованиям.

4. Интеграция и системное тестирование: объединение всех компонентов в единую систему и проверка корректности их совместной работы.

5. Развертывание и поддержка: предоставление готового продукта пользователям, установка датчиков WaterGuru Sense, обеспечение поддержки и обновлений системы.

6. Мониторинг и оптимизация: сбор отзывов от пользователей и анализ производительности системы с целью выявления возможных проблем и оптимизации работы компонентов.

**Часть 3.**

**Сопоставление функциональной и программной архитектуры:**

1. Мониторинг качества воды:

* Функциональный компонент: WaterGuru Sense
* Программные компоненты: SensorController, SensorData, WiFiManager, MeasurementData

2. Обработка данных и передача их на сервер:

* Функциональный компонент: Облачный сервер
* Программные компоненты: DataController, APIHandler, DatabaseManager

3. Получение данных с сервера и отображение их на мобильном устройстве:

* Функциональный компонент: Мобильное приложение
* Программные компоненты: WaterQualityData, UserData, NotificationManager, AdviceData

4. Управление уведомлениями и предоставление рекомендаций:

* Функциональный компонент: Мобильное приложение
* Программные компоненты: NotificationManager, AdviceData

**Сопоставление информационной архитектуры и архитектуры данных:**

1. Информационный объект "Данные о качестве воды":

* Структура данных: WaterQualityData
* Хранение: таблица в базе данных postgreSQL, связана с таблицей пользователей и датчиков

2. Информационный объект "Данные пользователя":

* Структура данных: UserData
* Хранение: таблица в базе данных postgreSQL, связана с таблицей данных о качестве воды и датчиков

3. Информационный объект "Данные прибора":

* Структура данных: SensorData
* Хранение: таблица в базе данных postgreSQL, связана с таблицей пользователей и данными о качестве воды

4. Информационный объект "Время измерения и показания":

* Структура данных: MeasurementData
* Хранение: встроенный в класс SensorData, передается на сервер и сохраняется в таблице данных о качестве воды

Организация хранения информационных объектов осуществляется в базе данных postgreSQL с использованием таблиц, связанных между собой через внешние ключи и индексы, обеспечивающие целостность данных и возможность их анализа и обработки.

Транзакционность выполнения операций, непротиворечивость и уникальность идентификаторов достигаются за счет использования системы баз данных postgreSQL 15, которая предоставляет функции транзакционности и управления целостностью данных. Запросы на изменение данных выполняются в единой транзакции и либо полностью применяются, либо отменяются в случае ошибки. Каждый объект в базе данных имеет уникальный идентификатор, который обеспечивает непротиворечивость и уникальность объектов.