**Промежуточный отчет по программному проекту**

**1. Основные планы и этапы проекта**

1.1 Краткое описание проекта:

Веб-приложение, предназначенное для автоматизации процесса обслуживания клиентов в ресторанах и кафе.

Название проекта: “Умный стол”

Цель проекта:

Создание системы для управления заказами в ресторанах с интеграцией через Telegram и удобной административной панелью для персонала

Краткое описание задач:

Разработать бэкенд для клиентской части приложения, а именно поддержать следующий функционал:

* 1. Ввод идентификатора стола с последующей инициацией сессии
  2. Вход в существующую сессию по идентификатору стола и коду сессии
  3. Запрос на удаление сессии
  4. Отображение каталога
  5. Вызов официанта
  6. Просмотр пользователей в сессии
  7. Просмотр содержимого заказа пользователя
  8. Добавление позиций в корзину
  9. Изменение количества позиции
  10. Просмотр содержимого корзины
  11. Редактирование содержимого корзины
  12. Просмотр детальной информации о позиции меню
  13. Добавление комментария к позиции
  14. Просмотр личного заказа
  15. Отображение статусов позиций в заказе
  16. Просмотр финальной информации по заказу

1.2 Планы и этапы выполнения проекта:

|  |  |  |  |
| --- | --- | --- | --- |
| Этап проекта | Описание работ | Ожидаемые результаты | Сроки выполнения |
| Техническое задание | Обоснование необходимости разработки, сбор данных, постановка задачи, выбор критериев эффективности. | Утвержденное техническое задание, определение требований и критериев для разработки бэкенда. | 01.11.24 – 15.11.24 |
| Разработка бэкенда | Разработка серверной части: создание API для сессий, корзины, меню, интеграции с Telegram и функционала для пользователей. | Рабочее API для всех функций: сессии, корзина, каталог, заказ, вызов официанта, интеграция с Telegram. | 16.11.24 – 31.03.25 |
| Разработка документации | Создание документации по API и архитектуре бэкенда, описание методов и функционала. | Полная техническая документация по проекту: описание всех эндпоинтов и их использования. | 01.04.25 – 15.04.25 |
| Испытание программы | Проведение тестирования, отладка программы, исправление ошибок. | Протестированные и отлаженные эндпоинты, исправленные баги, работающее приложение для сервера. | 16.04.25 – 30.04.25 |
| Внедрение | Подготовка программы и документации для защиты, защита проекта. | Готовая программа для защиты, успешная защита курсового проекта, загрузка материалов в ЛМС. | 01.05.25 – 15.05.25 |

**2. Используемый технологический стек и его обоснование**

2.1 Перечень используемых технологий:

|  |  |  |
| --- | --- | --- |
| Технология/Инструмент | Описание | Причины выбора |
| Go (Golang) | Язык программирования, ориентированный на высокую производительность и параллельную обработку. | Go выбран для разработки бэкенда из-за своей высокой производительности, простоты в использовании и отличной поддержки многозадачности, что критично для реального времени при обслуживании запросов от множества клиентов. |
| PostgreSQL | Реляционная база данных с открытым исходным кодом. | PostgreSQL был выбран за его надежность, поддержку транзакций и масштабируемость. Это позволяет эффективно обрабатывать и хранить большие объемы данных, такие как информация о заказах и сессиях. |
| Redis | База данных в памяти, используется для кэширования и хранения сессионных данных. | Redis выбран для кэширования данных и ускорения работы сессий, а также для хранения часто изменяющихся данных (например, корзины), что существенно повышает производительность приложения и уменьшает нагрузку на основную базу данных. |

2.2 Обоснование выбранного технологического стека:

Для разработки бэкенда веб-приложения был выбран стек технологий, который обеспечивает отличную производительность, масштабируемость и надежность, что крайне важно для решения задач, связанных с реальным временем и многозадачностью в ресторанном бизнесе.

Go (Golang) был выбран как основной язык разработки благодаря своей высокой производительности и возможностям для параллельной обработки запросов. Это особенно важно для нашего проекта, где приложение должно эффективно обрабатывать множество запросов одновременно, обеспечивая мгновенную реакцию на действия пользователей (например, добавление позиций в корзину или вызов официанта). Go идеально подходит для создания высокоскоростных сервисов, обладающих отличной производительностью и надежностью при высоких нагрузках.

PostgreSQL был выбран для хранения всех данных, связанных с пользователями, заказами и сессиями. Эта СУБД известна своей стабильностью и поддержкой транзакций, что критично для обеспечения целостности данных в системе. PostgreSQL также хорошо масштабируется и легко интегрируется с другими компонентами стека, что позволяет оптимально работать с большими объемами данных, что актуально для ресторанных сетей с большим количеством заказов и клиентов.

Redis используется для кэширования данных и хранения сессионной информации. Использование Redis позволяет значительно ускорить обработку запросов, связанных с пользовательскими сессиями и корзинами заказов, так как Redis хранит данные в памяти и предоставляет быстрый доступ к этим данным. Это помогает уменьшить нагрузку на основную базу данных и повысить общую скорость работы приложения.

Вместе эти технологии позволяют создать высокопроизводительное, масштабируемое и надежное веб-приложение, которое будет стабильно работать в условиях высоких нагрузок и обеспечивать быструю реакцию на действия пользователей.

**3. Критерии оценивания проекта**

|  |  |
| --- | --- |
| **Критерий** | **Описание** |
| Работа с базой данных | Будут использованы/Не будут использованы |
| Управление доступом | Будут использованы/Не будут использованы |
| Функциональность - Процент выполнения функциональных требований | Выполненные требования в процентах от общего количества |
| Функциональность - Количество реализованных функций | Абсолютное количество функций, которые работают правильно |
| Производительность и эффективность - Время отклика | Среднее время ответа программы на запросы пользователя (в секундах) |
| Качество кода - Средняя цикломатическая сложность | Средняя сложность функций |
| Тестирование - Процент успешных тестов (%) | Процент успешно пройденных тестов из общего количества |
| Соблюдение сроков и плана - Количество дней отклонения от плана | Общее число дней отклонения от плана |
| Использование технологического стека - Процент использования функциональности стека (%) | Процент использования функциональности выбранного стека технологий |
| Оценка командной работы - Среднее время коммуникации (в часах) | Среднее время, потраченное на обсуждение задач и решение вопросов |
| Оценка командной работы - Количество завершенных задач на каждого участника | Общее число задач, выполненных каждым членом команды |

**4. Особые пометки**

Разбиение задач между членами команды может быть не окончательным из-за того, что реализация каких-либо задач может занять больше/меньше времени чем было запланировано. Также возможно, что будет использован не весь предполагаемый стек (опасение насчёт Redis) и реализован не весь функционал, так как проект довольно большой и надо реализовать большое количество функционала, учитывая, что система должна быть распределённой и отказоустойчивой.