Индивидуальное описание проекта в рамках курсового проекта «Платформа для международных звонков через браузер»

Исполнитель: Н. Д. Смирнов, группа БПИ243
(Тимлид)

2025 г.

1. Основные планы и этапы проекта

1.1 Краткое описание проекта

- **Название проекта:** Платформа для международных звонков через браузер (Browser International Calls Platform).
- **Цель проекта:** Разработка веб-приложения для совершения международных аудиозвонков через браузер с использованием технологии WebRTC без необходимости установки дополнительного программного обеспечения.
- **Краткое описание задач (индивидуальная зона ответственности Н. Д. Смирнова):**
  1. **Архитектура и координация:**
     - Проектирование общей архитектуры системы (frontend, backend, взаимодействие компонентов).
     - Координация работы команды разработчиков.
     - Определение и документирование интерфейсов между компонентами.
     - Управление интеграцией компонентов, разработанных участниками команды.
  2. **Frontend разработка (структура и интеграция):**
     - Разработка структуры React-приложения (организация компонентов, страниц, утилит).
     - Реализация системы роутинга (React Router) между страницами: логин/регистра набор номера, история звонков.
     - Создание адаптивного веб-интерфейса для десктопных и мобильных устройств.
     - Интеграция компонентов, разработанных другими участниками: форма аутентификации (с Г. Д. Ворониным), компоненты управления звонком (с Г. Д. Ворониным), страница истории.
     - Реализация страницы истории звонков с отображением списка звонков (дата, время, номер, длительность).

- Обработка и отображение ошибок пользователю.
- Реализация базовой локализации интерфейса (RU/EN).

3. **Backend разработка (архитектура и интеграция):**
   - Проектирование общей архитектуры REST API.
   - Реализация базовой структуры сервера на Golang (роутер, middleware, структура проекта).
   - Интеграция API endpoints, разработанных другими участниками: аутентификация, управление звонками, история звонков.
   - Организация взаимодействия с базой данных (координация с Р. А. Николаевым).
   - Настройка обработки CORS и базовых middleware (логирование, обработка ошибок).
4. **Обеспечение качества и соответствия:**
   - Проверка соответствия реализации общему техническому заданию.
   - Обеспечение согласованности работы всех компонентов системы.

1.2 Планы и этапы выполнения проекта

Этап проекта	**Описание работ**	**Ожидаемые результаты**	**Сроки выполнения**
Архитектурное проектирование	Анализ требований, проектирование высокоуровневой архитектуры системы (frontend, backend, БД, внешние интеграции). Определение интерфейсов между компонентами.	Документ с архитектурой системы, диаграммы компонентов и последовательностей.	13.11.25 – 03.12.25
Организация проекта и координация	Создание репозитория, настройка среды разработки, распределение задач между участниками, установление процессов коммуникации.	Рабочий репозиторий, бэклог проекта, план работ.	15.11.25 – 16.12.25
Разработка базовой структуры Frontend	Настройка React-приложения, создание базовой структуры компонентов, реализация роутинга между основными страницами.	Каркас React-приложения с роутингом, основные layout-компоненты.	16.12.25 – 25.12.25
Разработка базовой структуры Backend	Настройка сервера на Golang, проектирование структуры API, реализация базовых middleware и обработчиков.	Работающий сервер с базовой структурой API, документация API.	25.12.25 – 31.01.26
Реализация адаптивного интерфейса	Разработка UI-компонентов с использованием адаптивного дизайна (десктоп + мобильные устройства). Создание страницы истории звонков.	Полный адаптивный интерфейс основных страниц, стилизованные компоненты.	01.02.26 – 10.02.26

Этап проекта	Описание работ	Ожидаемые результаты	Сроки выполнения
Интеграция компонентов команды	Интеграция frontend-компонентов (форма входа, управление звонком) и backend-endpoints (аутентификация, звонки, история) от других участников. Проверка взаимодействия.	Полностью интегрированное приложение с работающими сквозными сценариями.	10.02.26 – 17.02.26
Тестирование и отладка интеграции	Проведение интеграционного тестирования, отладка взаимодействия компонентов, проверка на различных браузерах и устройствах.	Стабильно работающее приложение, отчет о тестировании.	17.02.26 – 28.02.26
Документирование и подготовка к защите	Документирование архитектуры и кода, подготовка демонстрации, формирование итогового отчета.	Полная документация, готовый к защите проект.	01.03.26 – 15.03.26

2. Используемый технологический стек и его обоснование

2.1 Перечень используемых технологий

Технология/ Инструмент	Описание	Причины выбора
React	Библиотека для построения пользовательских интерфейсов.	Компонентный подход, высокая производительность, большое сообщество, позволяет создавать сложные интерактивные интерфейсы.
TypeScript	Надмножество JavaScript со статической типизацией.	Повышает надежность кода, облегчает рефакторинг и collaboration в команде, снижает количество runtime-ошибок.
React Router	Библиотека для маршрутизации в React-приложениях.	Стандартное решение для навигации в SPA, хорошо интегрируется с React.
CSS/SCSS с адаптивным дизайном	Технологии стилизации с поддержкой медиа-запросов.	Позволяет создать интерфейс, корректно отображающийся на устройствах с разными размерами экранов.
Golang	Язык программирования для backend-разработки.	Высокая производительность, простота разработки сетевых сервисов, эффективная работа с конкурентностью, сильная стандартная библиотека.

Технология/ Инструмент	**Описание**	**Причины выбора**
Gin/Gorilla Mux	Фреймворки/библиотеки для создания веб-серверов и API на Golang.	Упрощают создание REST API, предоставляют middleware, ускоряют разработку.
Git (GitHub/GitLab)	Система контроля версий.	Стандарт для командной разработки, позволяет эффективно управлять кодом, проводить code review, реализовывать CI/CD.
PostgreSQL	Реляционная система управления базами данных.	Надежность, богатые возможности, соответствие требованиям проекта по хранению структурированных данных (пользователи, звонки).
Docker	Платформа для контейнеризации приложений.	Обеспечивает консистентность окружений, упрощает развертывание и масштабирование.

2.2 Обоснование выбранного технологического стека

Выбранный стек технологий обеспечивает эффективную разработку, высокую производительность и поддерживает командную работу над проектом.

- **Frontend (React + TypeScript + React Router):** React является отраслевым стандартом для создания динамических веб-интерфейсов. Его компонентная архитектура идеально подходит для проекта, где разные участники разрабатывают независимые компоненты. TypeScript добавляет статическую типизацию, что критически важно для крупных проектов и командной разработки, так как позволяет выявлять ошибки на этапе компиляции и улучшает поддерживаемость кода. React Router обеспечивает четкую навигацию между страницами приложения.

- **Backend (Golang + Gin/PostgreSQL):** Golang выбран за его высокую производительность в сетевых приложениях и простоту разработки параллельных систем, что важно для обработки множества одновременных звонков. Фреймворк Gin ускоряет создание REST API и предоставляет необходимые middleware. PostgreSQL, как надежная и функциональная СУБД, обеспечивает целостность и сохранность данных.

- **Инструменты управления (Git, Docker):** Git является стандартом для контроля версий и позволяет команде эффективно работать над одним кодом. Docker обеспечивает одинаковое окружение для всех разработчиков и упрощает деплой.

- **Координация и интеграция:** Выбранные технологии имеют хорошую экосистему и документацию, что облегчает координацию работы команды. Четкое разделение на frontend и backend с определенными интерфейсами (REST API) позволяет участникам команды работать параллельно.

Данный стек является современным, масштабируемым и соответствует как техническим требованиям проекта, так и учебным целям, позволяя получить опыт работы с промышленными технологиями.

3. Критерии оценивания проекта

Для оценки индивидуального вклада Н. Д. Смирнова в проект предлагаются следующие количественные критерии:

Критерий	**Описание**	**Целевое значение**
Выполнение архитектурных требований	Процент реализованных архитектурных решений и выполненных задач по координации из индивидуального ТЗ.	100%
Качество frontend-архитектуры - Связность/зацепление модулей	Оценка степени связанности компонентов и модулей frontend-приложения (измеряется через метрики кода или экспертную оценку).	Низкая связанность, высокое зацепление
Качество backend-архитектуры - Время отклика API	Среднее время ответа основных API endpoints (в миллисекундах) при нагрузке.	< 200 мс
Полнота интеграции - Процент успешно интегрированных компонентов	Процент компонентов от других участников, успешно интегрированных в общую систему и работающих корректно.	100%
Качество документации - Полнота архитектурной документации	Процент описанных архитектурных решений, интерфейсов и процессов.	Не менее 90%
Соблюдение сроков координации - Процент проведенных встреч/синков	Протоколы встреч, своевременное решение блокирующих вопросов.	100%
Тестирование - Покрытие интеграционными тестами	Процент ключевых сценариев взаимодействия компонентов, покрытых интеграционными тестами.	Не менее 80%

4. Особые пометки

- Ключевой риск проекта — необходимость постоянной координации и синхронизации между участниками команды. Задержки у одного из разработчиков могут повлиять на график интеграции.
- Важным аспектом является четкое определение и соблюдение интерфейсов (API contracts) между frontend и backend, а также между различными backend-модулями.
- Успех интеграции компонентов, разрабатываемых разными участниками (WebRTC от Г. Д. Воронина, API истории звонков от Р. А. Николаева), напрямую зависит от качества и своевременности предоставления ими работающих модулей.
- Архитектурные решения должны быть достаточно гибкими, чтобы допускать изменения в реализации отдельных компонентов без переделки всей системы.
- Необходимо предусмотреть время на отладку кросс-браузерной совместимости и адаптивного дизайна, особенно для мобильных устройств.