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

Исполнитель: Г. Д. Воронин, группа БПИ243

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 Г. Д. Воронина):**
  1. Реализация WebRTC-клиента на стороне браузера (Frontend) для установления аудиосоединения.
  2. Разработка frontend-компонентов: выбор страны, ввод номера телефона, управление звонком (старт/стоп), отображение статуса звонка.
  3. Проверка доступа к микрофону и динамикам устройства пользователя.
  4. Обработка и отображение ошибок соединения.
  5. Интеграция на стороне сервера (Backend) с внешним VoIP API для установления звонков.
  6. Обработка WebRTC сигналинга (обмен SDP offer/answer).
  7. Реализация API endpoint'ов для инициализации (POST /api/calls/initiate) и завершения (POST /api/calls/terminate) звонков.
  8. Управление сессиями звонков.
  9. Интеграция с API для сохранения данных о звонках (взаимодействие с Р. А. Николаевым).
  10. Интеграция frontend-компонентов в общую структуру приложения (взаимодействие с Н. Д. Смирновым).

1.2 Планы и этапы выполнения проекта

Этап проекта	**Описание работ**	**Ожидаемые результаты**	**Сроки выполнения**
Научно-исследователь - ский	Изучение документации по WebRTC API, выбор внешнего VoIP сервиса, проектирование архитектуры взаимодействия frontend и backend.	Технический план реализации, список технологий.	13.11.25 – 03.12.25
Проектирование интерфейса и логики	Разработка макетов компонентов управления звонком. Проектирование схем обмена данными между frontend, backend и внешним API.	Макеты интерфейса, спецификация API endpoints, диаграммы последовательностей.	15.11.25 – 03.12.25
Разработка Frontend (WebRTC)	Реализация React-компонентов: панель набора номера, кнопки управления, индикатор статуса. Интеграция WebRTC API (getUserMedia, RTCPeerConnection).	Рабочие frontend-компоненты, установление локального аудиопотока.	16.12.25 – 25.12.25
Разработка Backend (Интеграция)	Реализация на Golang серверных handlers для инициализации/завершения звонков. Интеграция с выбранным VoIP API. Настройка передачи SDP данных.	Рабочие API endpoints, успешная интеграция с тестовым VoIP сервисом.	25.12.25 – 31.01.26
Интеграция и отладка	Соединение frontend и backend. Отладка установления полного аудиосоединения через браузер и внешний сервис. Обработка ошибок сети и API.	Полный рабочий цикл звонка: набор номера -> установление соединения -> разговор -> завершение.	01.02.26 – 10.02.26
Тестирование	Написание unit-тестов для критичных модулей. Проведение ручного тестирования сценариев звонка, проверка на разных браузерах и устройствах.	Отчет о тестировании, стабильная работа функционала.	10.02.26 – 17.02.26
Интеграция с общим проектом	Согласование интерфейсов и интеграция компонентов в общую codebase проекта (с Н. Д. Смирновым и Р. А. Николаевым).	Функционал звонков, работающий в общей платформе.	17.02.26 – 28.02.26
Подготовка к защите	Документирование кода, подготовка демонстрации, формирование отчета по индивидуальной части.	Готовый к демонстрации функционал, документация.	01.03.26 – 15.03.26

2. Используемый технологический стек и его обоснование

2.1 Перечень используемых технологий

Технология/ Инструмент	**Описание**	**Причины выбора**
JavaScript / TypeScript	Язык программирования для frontend-логики и работы с WebRTC API.	Стандарт для веб-разработки, полная поддержка WebRTC в браузерах. TypeScript добавляет статическую типизацию для надежности.
React	Библиотека для построения пользовательских интерфейсов.	Компонентный подход, высокая производительность, большое сообщество, согласовано с тимлидом для единства codebase.
WebRTC (Browser API)	Набор API (getUserMedia, RTCPeerConnection, RTCSessionDescription) для установления P2P аудиосоединений.	Нативный, мощный и стандартизированный браузерный API для передачи аудио/видео в реальном времени. Не требует плагинов.
Golang	Язык программирования для backend-части, отвечающей за интеграцию.	Высокая производительность, простота разработки сетевых сервисов, эффективная работа с конкурентными запросами, согласовано в команде.
HTTP / REST	Протокол и архитектурный стиль для взаимодействия frontend-backend и backend-external API.	Простота, универсальность, хорошая поддержка.
Git (GitHub/GitLab)	Система контроля версий.	Стандарт для командной разработки, позволяет отслеживать изменения, работать с ветками, интегрироваться с CI/CD.
VoIP API (напр., Twilio)	Внешний облачный сервис для маршрутизации международных звонков в телефонную сеть (PSTN).	Позволяет не разрабатывать сложную серверную телефонию с нуля, обеспечивает надежность и масштабируемость.

2.2 Обоснование выбранного технологического стека

Выбранный стек технологий оптимально соответствует задачам индивидуальной зоны ответственности и общим целям проекта.

- **Frontend (JS/TypeScript + React + WebRTC):** Позволяет создать интерактивный, отзывчивый интерфейс для управления звонком. WebRTC является отраслевым стандартом для коммуникаций в браузере, что гарантирует широкую поддержку и качество аудиосвязи. TypeScript повышает надежность кода за счет статической проверки типов, что критично для логики установления соединения.

- **Backend (Golang):** Отлично подходит для создания высокопроизводительных сетевых сервисов с малым временем отклика, что важно для обработки сигналинга WebRTC в реальном времени. Его эффективность и простота конкурентного программирования ускоряют разработку интеграции с внешними API.
- **Архитектура взаимодействия:** Использование HTTP/REST API для связи между компонентами обеспечивает четкое разделение ответственности, простоту отладки и возможность независимой разработки frontend и backend.
- **Интеграция с внешним сервисом (VoIP API):** Позволяет сосредоточиться на основной задаче — организации браузерной коммуникации — и не тратить ресурсы на разработку и поддержку сложной инфраструктуры для выхода в телефонные сети, что соответствует принципам экономической эффективности учебного проекта.

Данный стек является современным, востребованным на рынке и предоставляет все необходимые инструменты для успешной реализации поставленных задач по созданию стабильного и функционального модуля звонков.

3. Критерии оценивания проекта

Для оценки индивидуального вклада Г. Д. Воронина в проект предлагаются следующие количественные критерии:

Критерий	**Описание**	**Целевое значение**
Функциональность - Процент выполнения индивидуальных требований	Выполненные пункты технического задания (раздел 3.1 и 4.1 ТЗ Г. Д. Воронина) в процентах от общего количества.	100%
Качество кода - Покрытие кода тестами (%)	Процент строк кода в основных модулях (WebRTC-клиент, API handlers), покрытых модульными и интеграционными тестами.	Не менее 70%
Надежность - Количество критических ошибок (P0/P1) в основном сценарии	Количество ошибок, приводящих к невозможности установить или завершить звонок, обнаруженных в ходе приемочного тестирования.	0
Интеграция - Успешность интеграционных точек	Все запланированные интеграции (frontend-backend, backend-external VoIP API, backend-API истории звонков) работают корректно.	100% успешных интеграций
Соблюдение сроков - Процент выполнения задач в срок	Процент задач из индивидуального плана (раздел 1.2 данного описания), выполненных к установленным срокам.	Не менее 90%

4. Особые пометки

- Успех данной индивидуальной части напрямую зависит от своевременного получения доступа к внешнему VoIP API и его стабильной работы.

- Важным риском является возможная несовместимость WebRTC реализации в разных браузерах (особенно Safari), что требует дополнительного тестирования и, возможно, полифиллов.
- Реализация зависит от готовности общих интерфейсов и API, предоставляемых другими участниками проекта (Н. Д. Смирнов, Р. А. Николаев), что требует постоянной синхронизации.
- В случае изменения в выборе внешнего VoIP-сервиса может потребоваться корректировка backend-логики интеграции.