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

Исполнитель: Р.А. Николаев, 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 телефонных звонков через браузер с использованием технологии WebRTC, без необходимости установки дополнительного программного обеспечения.
- **Краткое описание задач (индивидуальная зона ответственности Р.А. Николаева):**
  - Разработка backend-части на Golang.
  - Реализация системы аутентификации и авторизации пользователей (JWT).
  - Проектирование и реализация схемы базы данных PostgreSQL.
  - Разработка API для работы с историей звонков (CRUD-операции).
  - Обеспечение безопасности данных (хеширование паролей, защита от SQL-инъекций, валидация).

1.2 Планы и этапы выполнения проекта

Этап проекта	**Описание работ**	**Ожидаемые результаты**	**Сроки**
Научно-исследовательский этап	Изучение технологий WebRTC, VoIP, аутентификации, проектирование схемы БД.	Определение архитектуры backend, выбор стека технологий.	13.11.25 – 03.12.25
Разработка backend (API)	Реализация эндпоинтов для аутентификации (/api/auth/register, /api/auth/login, /api/auth/logout).	Рабочие API для регистрации, входа, выхода.	25.12.25 – 31.01.26
Работа с базой данных	Создание таблиц users и calls, написание миграций, реализация запросов.	Готовая схема БД, оптимизированные запросы, CRUD для звонков.	25.12.25 – 31.01.26
Реализация модуля истории звонков	Разработка API для истории (/api/calls/history), пагинация, фильтрация.	API для получения, создания и обновления записей о звонках.	25.12.25 – 31.01.26
Обеспечение безопасности	Валидация данных, защита от SQL-инъекций, хеширование паролей (bcrypt), HTTPS.	Безопасный backend с защищёнными эндпоинтами.	25.12.25 – 31.01.26
Интеграция и тестирование	Интеграция с frontend, проведение функционального и нагрузочного тестирования API.	Протестированный backend, готовый к работе с frontend.	10.02.26 – 17.02.26
Подготовка к защите	Документирование кода, подготовка пояснительной записки и презентации.	Полный пакет документации, готовый проект.	01.03.26 – 15.03.26

2. Используемый технологический стек и его обоснование

2.1 Перечень используемых технологий

Технология/ Инструмент	**Описание**	**Причины выбора**
Golang	Язык программирования для backend-разработки.	Высокая производительность, простота конкурентности, богатая стандартная библиотека, подходит для масштабируемых сетевых приложений.
PostgreSQL	Реляционная система управления базами данных.	Надёжность, поддержка транзакций, богатый функционал, возможность сложных запросов, хорошая интеграция с Golang.
JWT (JSON Web Tokens)	Стандарт для создания токенов аутентификации.	Стандартный способ управления сессиями, не требует хранения состояния на сервере, подходит для RESTful API.
Gin / net/http	Фреймворк/библиотека для создания HTTP-сервера.	Gin предоставляет удобный роутинг и middleware, стандартная библиотека — минимализм и полный контроль.
bcrypt	Алгоритм хеширования паролей.	Устойчивость к brute-force атакам, адаптивная сложность, широкое признание в индустрии безопасности.
Git	Система контроля версий.	Стандарт в разработке, позволяет вести совместную работу, отслеживать изменения.
Docker (опционально)	Платформа для контейнеризации.	Обеспечение одинакового окружения для разработки и продакшена, упрощение развёртывания.

2.2 Обоснование выбранного технологического стека

Выбранный стек технологий ориентирован на создание надёжного, безопасного и производительного backend для веб-приложения.

Golang был выбран благодаря своей высокой производительности и эффективности в ра-

боте с сетевыми запросами, что критично для обработки вызовов в реальном времени. Его статическая типизация и строгая компиляция уменьшают количество runtime-ошибок.

PostgreSQL обеспечивает целостность данных, необходимую для хранения информации о пользователях и истории звонков, и хорошо интегрируется с Golang через драйверы (например, `pgx` или ORM GORM).

JWT упрощает реализацию аутентификации в stateless-архитектуре, что соответствует принципам REST.

Использование **bcrypt** для хеширования паролей соответствует современным стандартам безопасности.

Весь стек является популярным, имеет хорошую документацию и поддержку сообщества, что снижает риски и ускоряет разработку.

3. Критерии оценивания проекта

На основе общего ТЗ и индивидуальной зоны ответственности предлагаются следующие количественные критерии для оценивания проекта:

Критерий	**Описание**	**Целевое значение**
Функциональность — процент выполнения требований ТЗ	Доля реализованных функций из индивидуального ТЗ (аутентификация, БД, история звонков, безопасность).	100%
Качество кода — покрытие unit-тестами (%)	Процент строк кода backend (Golang), покрытых unit-тестами.	Не менее 70%
Качество кода — количество критических ошибок (Bugs/KLOC)	Количество обнаруженных критических ошибок на 1000 строк кода backend.	0
Надежность — время отклика API (мс)	Среднее время ответа основных API endpoints (регистрация, вход, получение истории) при нагрузке.	< 200 мс
Документация — полнота API документации	Наличие и полнота документации по API endpoints (например, в формате OpenAPI/Swagger).	100%

Таблица 3 – *Продолжение*

Критерий	**Описание**	**Целевое значение**
Соблюдение сроков — процент выполнения этапов в срок	Процент этапов из раздела 1.2, выполненных в запланированные сроки.	>90%

4. Особые пометки

- **Риски и зависимости:**
  - Зависимость от внешнего VoIP-сервиса для установки звонков (не входит в зону ответственности Р.А. Николаева, но требует координации по API).
  - Необходимость тесной интеграции с frontend-частью (координация с Н.Д. Смирновым и Г.Д. Ворониным по форматам JSON и endpoint'ам).
  - Риск неоптимальной производительности БД при увеличении объёма данных истории звонков — требует мониторинга и возможной дополнительной оптимизации.
- **Важные детали:**
  - Все пароли должны храниться исключительно в хешированном виде с использованием алгоритма bcrypt.
  - Необходимо обеспечить валидацию всех входящих данных как на клиенте, так и на сервере.
  - Все запросы к базе данных должны быть параметризованы для исключения SQL-инъекций.
  - Для работы в продакшене обязательна настройка HTTPS.
  - Ответственность за развёртывание backend-сервера и базы данных в тестовом/рабочем окружении согласуется с тимлидом (Н.Д. Смирнов).