Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка приложения для бесконтактной оплаты проезда в общественном транспорте»

по дисциплине «Проектный практикум»

Заказчик: Акмалетдинов Д.С. _____

Куратор: Харисов А.Р.
Доцент, кандидат технических наук _____

Студенты команды Острые пузырьки

Backend-разработчик: Антипин Н.И. _____

Frontend-разработчик: Узлов А.О. _____

Дизайнер: Малышева В.Е. _____

Аналитик: Пинькевич А.А. _____

Тимлид: Сулейменов А.Б. _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 Постановка проблемы и определение целевой аудитории 6
1.1 Проблема 6
1.2 Решение 6
1.3 Целевая аудитория 7
2 Анализ аналогов 8
2.1 Функциональный анализ конкурентов 8
2.2 Анализ способов оплаты проезда 9
3 Функциональные требования 10
4 Бэклог проекта 12
4.1 Этап аналитики 12
4.2 Этап разработки 12
4.2.1 1 неделя: 13
4.2.2 2 неделя: 13
4.2.3 3 неделя: 14
5 Архитектура программного продукта 15
5.1 Общая структура 15
5.2 Backend 15
5.2.1 Технологический стек: 15
5.2.2 Основные модули 16
5.3 Frontend 17
5.3.1 Технологический стек: 17
5.3.2 Архитектура клиентской части 17
5.4 Telegram-бот 18
5.5 Модели данных 18
5.6 Масштабируемость и перспективы развития 19
6 Методология и процесс разработки 20
7 Информация о работе каждого участника 22

ЗАКЛЮЧЕНИЕ 25
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27
ПРИЛОЖЕНИЕ А Результаты опроса на выявление ЦА 28
ПРИЛОЖЕНИЕ В Доска задач Yougile 30
ПРИЛОЖЕНИЕ С Пример функционального интерфейса 31
ПРИЛОЖЕНИЕ D Telegram-бот 32

ВВЕДЕНИЕ

Целью настоящего проекта является разработка Telegram-бота с интегрированным мини-приложением, предназначенного для упрощённой и бесконтактной оплаты проезда в общественном транспорте с использованием системы быстрых платежей (СБП). Основная задача проекта — предоставить пользователю удобный и безопасный способ получения ссылки для оплаты проезда, исключающий необходимость перемещения по салону транспортного средства или сканирования QR-кода, что особенно актуально в условиях высокой загруженности транспорта или при технических ограничениях мобильного устройства.

Актуальность проекта обусловлена растущей потребностью в цифровизации городских транспортных систем и повышении уровня комфорта и безопасности пассажиров. Современные реалии требуют от сервисов мобильности не только быстроты и надёжности, но и доступности, в том числе для пользователей, не имеющих возможности использовать традиционные способы оплаты. Разработка такого программного продукта способствует улучшению пользовательского опыта и снижению нагрузки на традиционные системы оплаты, одновременно предлагая более гибкие и инклюзивные решения для пассажиров.

Область применения разработанного программного продукта охватывает сферу городского транспорта. Бот будет востребован среди пассажиров, которые регулярно пользуются общественным транспортом и стремятся к более удобному и технологичному способу взаимодействия с системой оплаты. Также он может быть использован транспортными компаниями и муниципалитетами в рамках цифровизации городской инфраструктуры.

Ожидаемыми результатами проекта являются: создание полностью работоспособного Telegram-бота с мини-приложением, реализующего три варианта бесконтактной оплаты по СБП (по номеру маршрута/ТС, по локации пользователя и по остановкам на карте), а также функции сохранения маршрутов в избранное, просмотра истории поездок и получения уведомлений о приближении избранного транспорта. Кроме того, пользователю будет доступен поиск маршрута для оплаты, управление уведомлениями и возможность обращения в техническую поддержку. Реализация всех перечисленных функций запланирована в течение одного семестра.

1 Постановка проблемы и определение целевой аудитории

1.1 Проблема

В процессе проведения опроса среди студентов и молодых людей(), регулярно пользующихся общественным транспортом, была выявлена ключевая проблема — неудобство и нестабильность оплаты проезда по QR-коду, несмотря на его широкое распространение.

Наиболее частые трудности при оплате QR-кодом:

– Камера не фокусируется или долго распознаёт код;

– QR-код недоступен: закрыт другими пассажирами, повреждён или размещён в неудобном месте;

– Перебои с интернет-соединением или сбои в работе платёжных сервисов;

– Сложность подойти к нужному месту в салоне транспорта;

– Долгое время отклика сайта после перехода по ссылке.

Несмотря на повсеместное внедрение QR-систем, пользовательский опыт остаётся неудовлетворительным, особенно в часы пик, при плохом освещении или в движущемся транспорте. Это создаёт барьеры для быстрой и комфортной оплаты, особенно для молодёжи, привыкшей к мобильным сервисам.

1.2 Решение

Исходя из собранных данных, целесообразно внедрение альтернативной и более удобной системы — оплаты проезда через Telegram-бот с mini app, в котором можно:

– Оплатить проезд по номеру маршрута;

– Сохранять избранные маршруты;

– Получать уведомления о начале движения и прибытии нужного транспорта;

– Сохранять историю поездок.

Такой подход решает ключевые проблемы, устраняя зависимость от физических QR-кодов и минимизируя необходимость взаимодействия с неудобными внешними элементами транспорта.

1.3 Целевая аудитория

Целевой аудиторией проекта являются:

– Студенты и молодые специалисты, которые активно пользуются общественным транспортом;

– Владельцы смартфонов, активно использующие мессенджер Telegram (по результатам опроса — почти 100% респондентов);

– Пассажиры, предпочитающие цифровые и бесконтактные способы оплаты;

– Люди, испытывающие трудности при текущей системе оплаты проезда (QR/Банковская карта/Екарта).

Анализ показал, что подавляющее большинство участников опроса открыто к использованию Telegram-бота для оплаты и считают функцию ввода номера маршрута удобной. Также многие заинтересованы в получении уведомлений о движении транспорта, если они будут реализованы гибко и с возможностью отключения (приложение А).

2 Анализ аналогов

2.1 Функциональный анализ конкурентов

Таблица 1 — Функциональный анализ конкурентов

Функциональность	**Пример у конкурентов**	**Описание функции**	**Интеграция в проект**	**Примечания и технологии**
Запрос геолокации / выбор города	Moovit, GorodPay	При открытии мини-приложения система запрашивает геолокацию; при отказе — выбор вручную	Да	Использование API геолокации, fallback на ручной выбор через выпадающий список
Поиск маршрутов по номеру / ТС	Avtobys, Moovit	Строка поиска с автодополнением по номеру маршрута, отображение направления и остановки	Да	React-инпут с автодополнением, API для подгрузки маршрутов
Страница оплаты проезда	GorodPay, «Оплати»	Отдельный экран с отображением способа оплаты и ссылкой на оплату	Да	Модальное окно или экран с кнопкой «Оплатить проезд»
Генерация QR-кода	«Оплати»	Генерация QR-кода после оплаты для визуального подтверждения	Да	Использование библиотеки qrcode.react, модальное окно с возможностью закрытия
Поделиться ссылкой на оплату	Идея новая, у конкурентов отсутствует	Возможность отправить ссылку на оплату другим пользователям через Telegram	Возможна	Telegram Web Apps API: генерация ссылки и переход в чат
Добавление маршрута в избранное	GorodPay	Кнопка добавления маршрута в избранное (сердечко) для быстрого доступа	Да	Redux/Zustand или localStorage для хранения состояния избранного маршрута

2.2 Анализ способов оплаты проезда

Таблица 2 — Анализ способов оплаты проезда

Критерий	QR-коды через СБП	Терминалы (карта/NFC)	Городские транспортные приложения	Наш Telegram-бот
Как работает?	Скан QR-кода → переход в банк → выбор банка → оплата	Прикладывание карты/телефона к валидатору	Установка приложения → регистрация → покупка билета → QR	Ввод номера маршрута в Telegram → оплата через ссылку СБП
Необходимость физического контакта	Да	Да	Нет	Нет
Удобство в переполненном транспорте	Плохо — до QR сложно дотянуться	Хорошо — достаточно приложить	Хорошо	Отлично — оплата возможна в любом месте из Telegram
Требуется установка приложения	Нет	Нет	Да	Нет (если Telegram уже установлен)
Интернет-соединение	Да	Нет	Да	Да
Масштабируемость (города)	Ограничено — новые QR в каждом городе	Ограничено — нужны терминалы	Часто локальные	Да — легко масштабируется через конфигурацию бэкенда
Стоимость внедрения	Низкая — печать QR-кодов	Высокая — терминалы, валидаторы	Высокая — разработка, поддержка	Низкая — разработка Telegram-бота и генерация ссылок
Поддержка способов оплаты	Через СБП	Банковские карты, NFC	Банковские карты, счета, подписки	Через СБП

На фоне существующих решений Telegram-бот предлагает более простой, доступный и масштабируемый способ оплаты, особенно актуальный для молодёжи и активных пользователей мессенджеров. Отсутствие необходимости установки нового приложения и физического контакта делает его удобным решением в условиях современного транспорта.

3 Функциональные требования

В рамках разработки Telegram-бота с мини-приложением для оплаты проезда были проанализированы основные сценарии использования, основанные на требованиях заказчика и потребностях пользователей. Ниже приведено подробное описание функциональных требований, отражающих предполагаемое поведение системы и действий пользователя.

Таблица 3 — Основной пользовательский сценарий

Действие пользователя	Реакция системы
Заходит в бота и нажимает стартовую кнопку / пишет /start	Отображается стартовое сообщение с описанием возможностей бота и кнопками для взаимодействия
Нажимает на кнопку открытия мини-приложения	Система запрашивает доступ к геолокации
Предоставляет / не предоставляет доступ к геолокации	Если доступ получен — подгружается ближайший транспорт, если нет — появляется окно выбора города вручную
Выбирает город вручную	Система сохраняет город как выбранный по умолчанию для будущих сессий

Таблица 4 — Оплата проезда по номеру маршрута / транспорта

Действие пользователя	Реакция системы
Вводит номер маршрута / ТС	Подгружаются совпадения с указанием направления и текущей остановки
Выбирает нужный маршрут	Загружается страница оплаты
Нажимает "Оплатить проезд"	Появляется основная и дополнительные ссылки на оплату через СБП
Подтверждает успешную оплату	Система закрывает окно оплаты
Отмечает, что оплата не прошла	Система предлагает другую ссылку и уведомляет об ошибке

Таблица 5 — Работа с избранными маршрутами

Сценарий	Реакция системы
Открытие "Избранное"	Загружается список сохранённых маршрутов
Выбор маршрута	Переход к оплате (тот же сценарий, что в пункте 3.2)
Добавление маршрута через "+"	Предлагается форма выбора маршрута и настройки уведомлений (остановка, направление, дни и время)
Редактирование маршрута	Открывается окно редактирования параметров уведомлений
Удаление маршрута	Система запрашивает подтверждение и удаляет маршрут

Таблица 6 — Поиск транспорта по геолокации и карте

Сценарий	Реакция системы
Нажатие на "Транспорт рядом"	Если доступна геолокация — отображаются ближайшие ТС
Выбор ТС	Загружается страница оплаты (см. 3.2)
Выбор остановки на карте	Отображаются ТС, которые недавно были или скоро прибудут

Сценарии использования бота строятся вокруг быстрого доступа к оплате проезда и управления маршрутами без необходимости установки отдельных приложений. Система учитывает как автоматическое определение местоположения, так и ручной выбор города, предлагает удобные способы оплаты, функции избранного, уведомлений и возможность делиться ссылками на оплату. Всё это делает Telegram-бот гибким и масштабируемым решением для внедрения в различных городах.

4 Бэклог проекта

Работа над проектом была разделена на два ключевых этапа: аналитический и этап разработки. Каждый этап включал в себя распределение задач между участниками команды с учётом их ролей и компетенций.

4.1 Этап аналитики

На этом этапе основное внимание было уделено исследованию, планированию и подготовке:

– Пинькевич Артём

Составление схемы пользовательского потока (User-Flow) и проведение анализа конкурентов с точки зрения логики работы и структуры приложений;

– Малышева Вероника

Разработка предварительных набросков и эскизов интерфейса мини-приложения для формирования общей визуальной концепции;

– Узлов Андрей

Проведение технического анализа конкурентов, фокус на фронтенд-решениях, применяемых в аналогичных сервисах;

– Сулейменов Артур

Составление дорожной карты проекта с разбивкой этапов разработки и ключевыми точками контроля выполнения задач;

– Антипин Николай

Формирование всех основных сценариев использования приложения с описанием действий пользователя и откликов системы.

4.2 Этап разработки

Разработка была разбита на три недели, каждая из которых имела конкретные цели и задачи:

4.2.1 1 неделя:

– Пинькевич Артём

Проверка текущей реализации, выявление недочетов и составление отчета по доработкам;

– Узлов Андрей

Настройка маршрутизации в приложении, подключение библиотек, вёрстка главного экрана, вкладок "История" и "Выбор города";

– Сулейменов Артур

Подключение базы данных PostgreSQL, реализация моделей (пользователи, маршруты и прочее);

– Антипин Николай

Разработка парсера данных: маршруты, остановки, QR-коды;

– Малышева Вероника

Полная реализация главного экрана, выбора города и вкладки "История".

4.2.2 2 неделя:

– Пинькевич Артём

Проведение ревизии выполненных задач, составление отчета, дополнение списка доработок;

– Узлов Андрей

Вёрстка вкладки "Избранное", страницы транспортного средства (оплата), подключение API для оплаты и истории проезда;

– Сулейменов Артур

Реализация API для блока "Оплата проезда";

– Антипин Николай

Написание всех обработчиков в Telegram-боте, реализация API для вкладки "История проезда", подключение уведомлений;

– Малышева Вероника

Реализация страницы транспортного средства (оплата) и вкладки "Избранное".

4.2.3 3 неделя:

– Пинькевич Артём

Документирование всех API;

– Узлов Андрей

Вёрстка вкладки "Карта", подключение оставшихся API, реализация вкладки "Поддержать авторов";

– Сулейменов Артур

Разработка API для вкладки "Карта" и главной страницы;

– Антипин Николай

Реализация API для вкладки "Избранное";

– Малышева Вероника

Финальная вёрстка вкладки "Карта" и вкладки "Поддержать авторов", реализация адаптивного дизайна под планшеты.

5 Архитектура программного продукта

5.1 Общая структура

Программный продукт построен по принципам клиент-серверной архитектуры, включающей следующие ключевые компоненты:

– Frontend — веб-интерфейс и мини-приложение Telegram, реализованные на React;

– Backend — серверная часть на Django с использованием Django Ninja для построения API;

– База данных — PostgreSQL, развернутая в Docker-контейнере;

– Парсеры данных — отдельные модули для сбора информации о маршрутах, транспорте и остановках из сторонних источников (Bustime, 2ГИС и др);

– Telegram-бот — реализован для взаимодействия с пользователем через команды, инлайн-режим и уведомления;

Приложение использует Clean Architecture, что обеспечивает модульность, простоту масштабирования и поддержку проекта в долгосрочной перспективе.

5.2 Backend

5.2.1 Технологический стек:

– Python;

– Django + Django Ninja;

– Django ORM;

– PostgreSQL;

– Docker.

5.2.2 Основные модули

Модуль парсинга данных:

– Получение маршрутов;

– Получение остановок;

– Получение транспорта в определённом квадрате координат;

– Получение транспорта, приближающегося к остановке;

– Экспериментальные методы (данные получаются, но ещё не используются в логике).

Модуль обработки QR-кодов:

– Извлечение информации (тип транспорта, маршрут, номер ТС, QR-код);

– Сохранение во временное хранилище (текстовый файл).

API для клиентской части:

– Работа с избранными маршрутами;

– История поездок;

– Уведомления;

– Список городов, маршрутов, транспорта и остановок.

Интеграция с Telegram:

– Inline-режим поиска транспорта и отправки ссылок на оплату;

– Поддержка, помощь, управление уведомлениями;

– Обратная связь через Telegram-канал.

Работа с локацией:

– Выбор города;

– Масштабируемое API, не зависящее от конкретного города (Bustime + 2ГИС).

5.3 Frontend

5.3.1 Технологический стек:

– JavaScript;

– React, MobX, React Router DOM, react-select, react-yandex-maps;

– CSS-переменные, адаптивная верстка, поддержка тёмной темы.

5.3.2 Архитектура клиентской части

Layout — отображение разных элементов в зависимости от маршрута

MobX store — централизованное хранилище данных о транспорте, пользователе, истории, избранном

Маршрутизация — реализована через react-router-dom

Темизация — реализована с помощью Context API и custom hook useTheme

Страницы(приложение С):

– Главная
– Карта ближайших остановок
– Выбор геолокации
– История поездок
– Оплата проезда
– Избранное и управление уведомлениями
– Страница поиска транспорта

Компоненты:

– Универсальные (Header, Footer, SearchBar)
– Карточки транспорта, маршрутов, остановок
– Блоки информации (IntroBlock и др)
– Карта и фильтрация остановок

Интеграция с API: структура сторов адаптирована под backend API

5.4 Telegram-бот

Telegram-бот интегрирован с основным приложением и расширяет его функциональность. Он обрабатывает базовые команды, позволяет:

– получать ссылку на оплату проезда с помощью inline-мода;

– включать/отключать уведомления;

– обращаться в поддержку;

– открывать мини-приложение по кнопке.

Сообщения пользователей в поддержку автоматически отправляются в приватный канал, что повышает скорость реакции. Также бот может выполнять ряд функций без открытия мини-приложения, что удобно в реальных условиях(приложение Д).

5.5 Модели данных

Для корректной работы API и бизнес-логики приложения была разработана и реализована структура моделей, включающая:

– Пользователь (User) — хранит информацию о пользователе, включая ID Telegram и настройки;

– Транспортное средство (Transport) — содержит данные об общественном транспорте: номер, тип, текущие координаты и т.д.;

– Маршрут (Route) — включает описание маршрута, его уникальный ID, направления и связанные остановки;

– Направление маршрута (RouteDirection) — определяет конкретное направление движения на маршруте;

– Остановка (Stop) — содержит географические координаты, тип и привязку к маршрутам;

– Уведомления (Notification) — настройки уведомлений пользователя по конкретному маршруту;

– Избранное (Favourites) — список избранных маршрутов, выбранных пользователем;

– История поездок (Ride) — информация о совершённых пользователем поездках.

Модель QR-кодов, содержащая тип транспорта, номер маршрута, ТС и ссылку на оплату, пока не реализована в БД, но подготовлена и временно сохраняется в текстовом виде. Эта модель будет добавлена после финального этапа парсинга.

5.6 Масштабируемость и перспективы развития

Одной из важных задач на этапе проектирования была возможность масштабирования проекта под новые города и транспортные системы. Изначально данные получались с сайтов городских транспортных департаментов, однако этот подход оказался ограничен: подобные сайты есть только в крупных городах.

В процессе разработки был найден альтернативный источник — Bustime и 2ГИС, предоставляющие необходимые методы и охватывающие почти все крупные города России. Это позволило выстроить универсальную архитектуру парсинга, которая не зависит от конкретного города.

Кроме того, проект планируется перевести на асинхронный режим работы и, при необходимости, внедрить кэширование данных, что значительно ускорит ответы API и снизит нагрузку на сервер. Деплой проекта на сервер также включён в финальный этап подготовки к защите.

6 Методология и процесс разработки

В рамках разработки проекта была выбрана методология Agile, ориентированная на гибкую и поэтапную реализацию функционала. Работы велись итерационно: каждый этап включал в себя планирование задач, реализацию функциональности и предоставление результатов в виде отчетов. Процесс разработки был интегрирован в дисциплину "Проектный практикум", что также предусматривало регулярную отчетность по итерациям перед куратором проекта.

Командная работа организовывалась в формате еженедельных спринтов, по результатам которых каждый участник формировал индивидуальный отчет. Контроль выполнения задач осуществлялся тимлидом, который вел доску задач в системе YouGile(приложение Б). Распределение задач по команде также происходило через данную систему.

В команде были четко разграничены роли: за backend-часть отвечал один участник (в том числе реализацию API и логики Telegram-бота), frontend-часть мини-приложения была реализована другим участником, также в проекте участвовали дизайнер, подготовивший макеты в Figma, и аналитик, отвечавший за постановку задач и обработку пользовательских данных, собранных через опрос.

Коммуникация между членами команды осуществлялась преимущественно через мессенджер Telegram, а также проводились регулярные еженедельные созвоны в Microsoft Teams для синхронизации и планирования дальнейших шагов.

Исходный код проекта хранился и велся в распределенной системе контроля версий Git с размещением в GitHub-репозиториях. Это обеспечивало удобный доступ для всех членов команды, возможность одновременной работы над проектом и контроль версий. Макеты интерфейсов и дизайн-проекты хранились и согласовывались через Figma, что позволяло удобно просматривать и адаптировать элементы интерфейса под текущие потребности проекта.

Таким образом, применяемая методология и организация командного взаимодействия позволили эффективно реализовать проект в сжатые сроки, с учётом специфики командной разработки и учебного формата.

7 Информация о работе каждого участника

В рамках реализации программного продукта была организована командная работа, в которой каждый участник проекта выполнял определённые задачи, соответствующие его роли. Слаженность действий и регулярное взаимодействие между участниками обеспечили эффективное продвижение проекта на всех его этапах.

Аналитик осуществил ряд ключевых задач, направленных на обоснование необходимости создания продукта и построение логики пользовательского взаимодействия:

– Разработана карта пользовательского пути (User-Flow), которая легла в основу проектирования интерфейса и алгоритма работы с системой;

– Выполнен сравнительный анализ аналогичных решений, представленных на рынке. На основе анализа определены сильные и слабые стороны конкурентов, что позволило выстроить конкурентоспособную и удобную архитектуру пользовательского взаимодействия;

– В ходе реализации проекта аналитик составлял регулярные отчёты по результатам своей деятельности, что способствовало прозрачности всех процессов и своевременному внесению корректив;

– Был проведен сбор обратной связи от пользователей, который стал важным источником информации для финальной доработки проекта и повышения его удобства.

Дизайнер отвечал за визуальную составляющую мини-приложения:

– Проведён анализ дизайнерских решений в других мини-приложениях Telegram и мобильных интерфейсах. Это позволило выделить лучшие практики и применить их в разрабатываемом продукте;

– Созданы детализированные макеты всех ключевых страниц, включая: главную страницу, экран выбора города, историю поездок, экран оплаты, избранное и карту. Отдельно реализована поддержка тёмной темы;

– Дизайн был адаптирован под планшеты, что обеспечило кроссплатформенность интерфейса;

– Вся дизайн-система реализована по принципу атомарного и молекулярного подхода, что обеспечило визуальную целостность и удобство масштабирования интерфейсов.

Frontend-разработчик осуществил реализацию клиентской части:

– Инициировал и организовал репозиторий проекта на GitHub. Провёл анализ архитектуры клиентской части конкурентов для выбора наиболее эффективных решений;

– Реализовал интерфейсы всех основных страниц: главная, выбор города, история поездок, страница оплаты, избранное, карта. Также внедрена поддержка тёмной темы;

– В процессе разработки выполнен рефакторинг структуры MobX с учётом логики взаимодействия с backend-частью, что повысило стабильность и надёжность системы управления состоянием;

– Проведена интеграция с Telegram API, развернуто окружение в Docker-контейнере, что упростило локальную разработку и тестирование.

Backend-разработчик отвечал за разработку серверной логики:

– Создан основной репозиторий backend-части с реализацией принципов «чистой архитектуры», разработаны сценарии использования, легшие в основу логики системы;

– Реализован парсер, автоматически извлекающий необходимые данные для функционирования продукта;

– Разработан Telegram-бот, способный обрабатывать обращения пользователей, в том числе — предоставлять ссылку для оплаты по номеру маршрута или транспортного средства;

– Реализованы API-эндпоинты, обеспечивающие работу функций «Избранное» и «История поездок». Выполнен деплой проекта: зарегистрирован домен, настроена VPS, установлены nginx и Cloudflare для обеспечения безопасности и стабильности.

Тимлид координировал все этапы разработки:

– Организовал процесс командной работы через систему управления проектами Yougile, где велось регулярное планирование задач и контроль исполнения;

– Провёл исследование целевой аудитории посредством анкетирования, результаты которого были использованы для определения функционального ядра приложения;

– Спроектировал и реализовал структуру базы данных, настроил отображение сущностей в административной панели;

– Разработал API-эндпоинты для страниц «Оплата», «Карта» и «Выбор города».

ЗАКЛЮЧЕНИЕ

Разработанный программный продукт в полной мере соответствует требованиям, сформулированным на этапе постановки задачи. Цель проекта — создание Telegram-бота и мини-приложения для упрощения оплаты проезда и отслеживания маршрутов общественного транспорта — была достигнута. На основе анализа пользовательских потребностей и результатов опроса были выделены ключевые функции, в том числе: выбор города, поиск маршрутов, оплата, история поездок, добавление маршрутов в избранное, отображение на карте и уведомления. Все эти функции были реализованы и протестированы, что подтверждает соответствие продукта ожиданиям как заказчика, так и конечных пользователей.

Качество продукта было проверено путём ручного тестирования. По его результатам были выявлены отдельные сложности, не влияющие на стабильность основной логики приложения. Одним из затруднений стало ограничение функционала стороннего API Bustime, из-за чего пришлось использовать 2ГИС в качестве альтернативы. Также сложности возникли при проектировании логики взаимодействия с внешними API — не всегда было очевидно, какие именно данные нужно передавать и получать, что потребовало дополнительного времени на проработку. Несмотря на эти трудности, все основные задачи были успешно решены, и продукт продемонстрировал стабильную и корректную работу.

В дальнейшем проект может быть улучшен за счёт усиления мер безопасности — в частности, внедрения механизмов защиты от несанкционированного доступа к API со стороны внешних пользователей. Это повысит надёжность системы и защитит данные от потенциальных угроз. Также возможно дальнейшее расширение функциональности и поддержка большего числа городов по мере масштабирования проекта.

В целом, продукт можно охарактеризовать как устойчивый, функциональный и отвечающий современным требованиям удобства,

надёжности и доступности. Грамотно выстроенная командная работа, применение гибкой методологии и ориентация на реальные пользовательские потребности обеспечили успешную реализацию поставленных целей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство по Telegram Bot API [Электронный ресурс] / Telegram. – URL: https://core.telegram.org/bots/api

2. Руководство по созданию мини-приложений Telegram [Электронный ресурс] / Telegram. – URL: https://core.telegram.org/bots/webapps

3. Документация 2ГИС API [Электронный ресурс] / ООО «2ГИС». – URL: https://docs.2gis.com/ru

4. GitHub-репозиторий Bustime [Электронный ресурс]. – URL: https://github.com/bustime-org/bustime.git

5. Официальная документация по Django REST Framework [Электронный ресурс] / Encode OSS Ltd. – URL: https://www.django-rest-framework.org/

6. Документация по Django Ninja [Электронный ресурс] / Vitalik. – URL: https://django-ninja.dev/

7. Документация Telegram Web Apps JS API [Электронный ресурс] / Telegram. – URL: https://core.telegram.org/bots/webapps#initializing-mini-apps

8. GitHub – официальный сайт хостинга исходного кода и системы контроля версий [Электронный ресурс]. – URL: https://github.com/

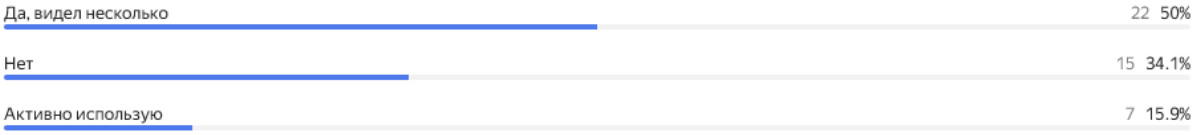
9. Документация YouGile: Система управления проектами [Электронный ресурс] / ООО «Югайл». – URL: https://yougile.com/ru/help

ПРИЛОЖЕНИЕ А

Результаты опроса на выявление ЦА

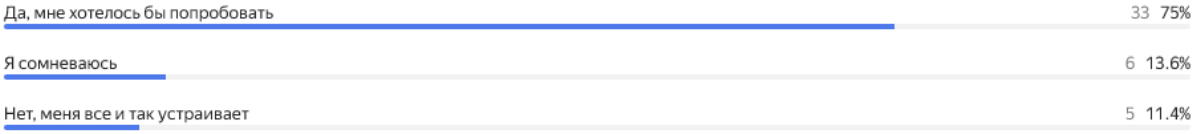
Знакомы ли Вы со встроенными приложениями Телеграмм? (миниапп)

44 ответа



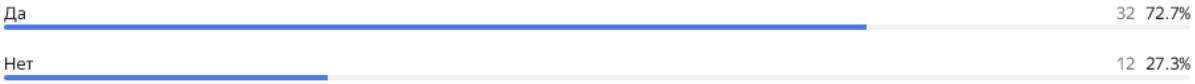
Было бы Вам удобно оплачивать проезд в общественном транспорте через Телеграмм, не перемещаясь по салону?

44 ответа



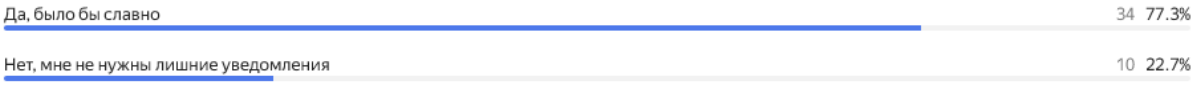
Было бы Вам удобно получать ссылку на оплату, вводя номер общественного транспортного средства?

44 ответа



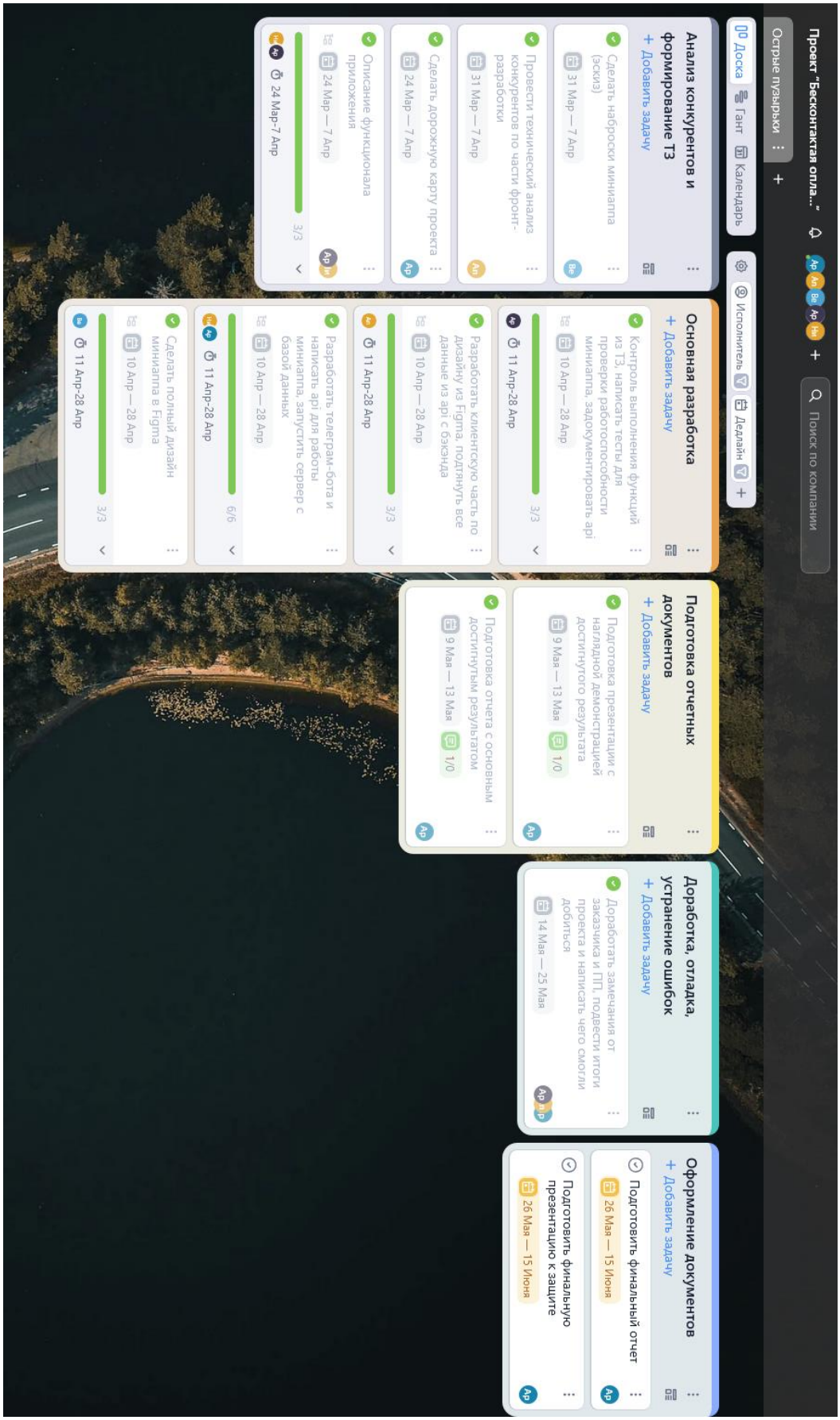
Хотели ли бы Вы заранее получать уведомление о приближающемся общественном транспорте, которым вы регулярно пользуетесь?

44 ответа



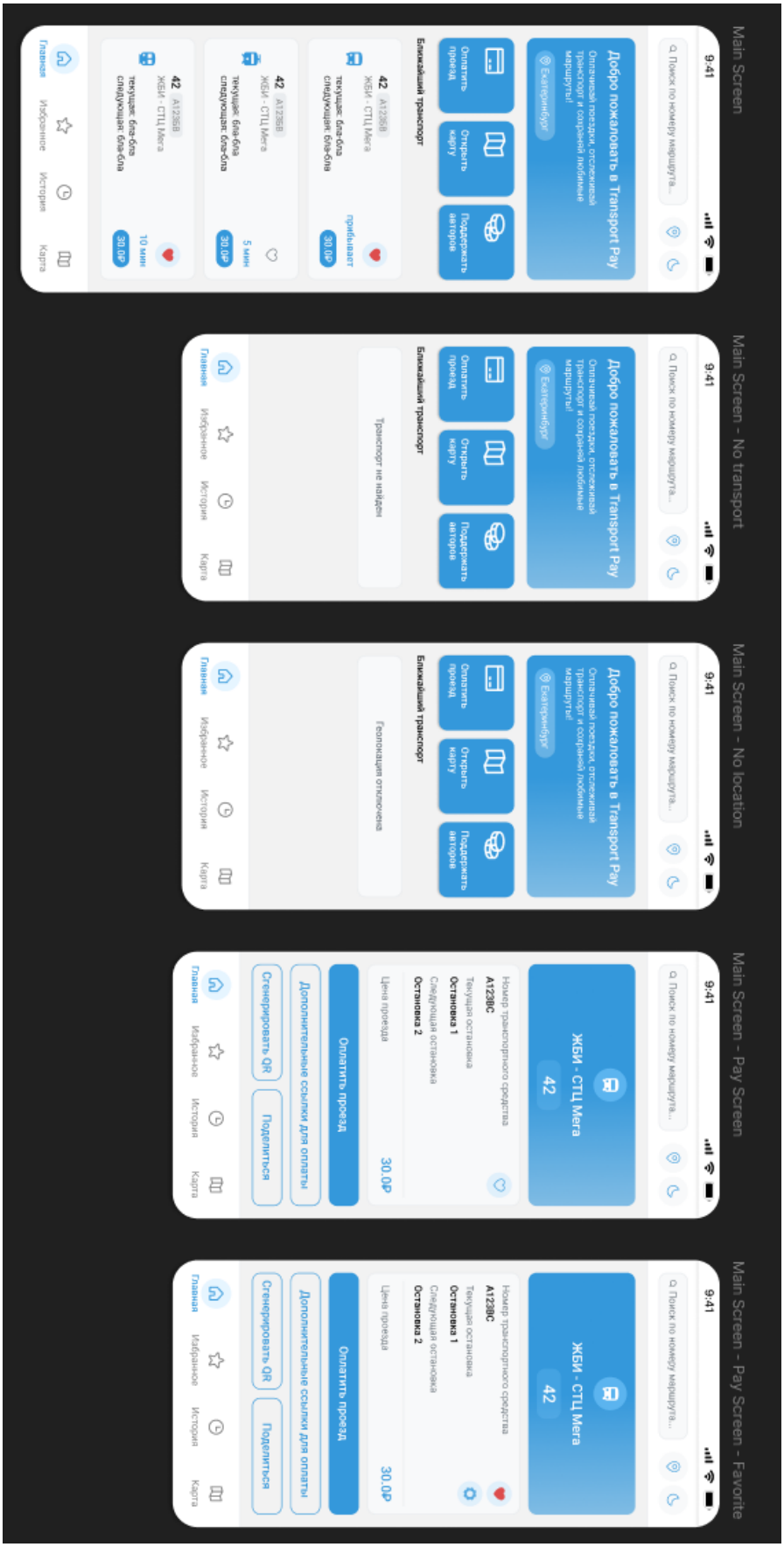
ПРИЛОЖЕНИЕ В

Доска задач Yougile



ПРИЛОЖЕНИЕ С

Пример функционального интерфейса



ПРИЛОЖЕНИЕ D

Telegram-бот