

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Исследование предметной области

1.1.1 Клиент-серверная архитектура системы

«Клиент-сервер» - архитектура, которая чаще всего ложится в основу проектирования современных многопользовательских систем. Ее использование позволяет разделить обязанности и ответственности между клиентами и сервисом. Проектирование разрабатываемой в дипломном проекте системы будет осуществляться в соответствии с клиент-серверной архитектурой.

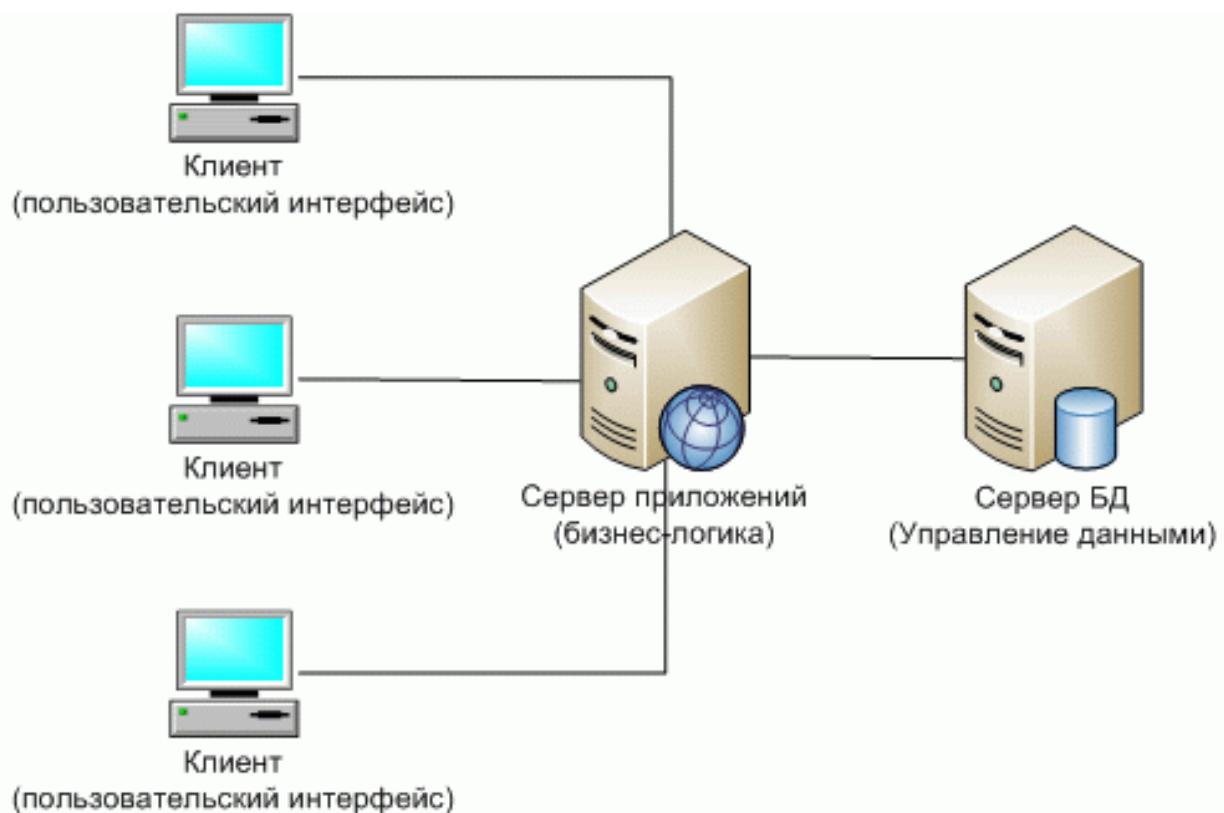


Рисунок 1.1 – Схема клиент-серверной архитектуры

Клиент-серверная архитектура заключает в себе идею использования одной или нескольких машин для создания одного продукта/системы, разделяя обязанности между частями этого продукта/системы. Разделение обязанностей достигается за счет деления системы на несколько программных модулей – клиентское ПО и серверное ПО. Обычно клиентское и серверное ПО взаимодействуют друг с другом посредством общения между устройствами системы с использованием различных сетевых протоколов, например HTTP, FTP, RPC и т.д., однако они также могут находиться и на

одном устройстве, используя для общения внутренние протоколы операционной системы, на которой они выполняются.

Данная архитектура, в отличие от peer-to-peer, является централизованной, то есть конечные пользователи (клиенты), как и работоспособность всей системы, зависят от центральной точки – сервера, что является как преимуществом, так и недостатком.

Преимущества данной архитектуры:

- необходимость в наличии мощного устройства только со стороны сервера, клиентом же могут выступать современные устройства любой мощности, от персональных компьютеров до умных часов и смартфонов;
- необходимость обеспечивать защиту и хранить пользовательские данные только на стороне сервера ввиду того, что хранить их на устройстве конечного пользователя небезопасно;
- отсутствие дублирования кода между различными программными компонентами системы, что позволяет сделать клиенты легкими и быстрыми даже на самых слабых устройствах;

Недостатки данной архитектуры:

- отказ серверной части архитектуры вызывает неработоспособность всей системы;
- высокая стоимость серверного оборудования ввиду высоких требований к его производительности;
- поддержка серверного оборудования требует выделенных специалистов;

В данном случае применение данной архитектуры целесообразно ввиду непостоянного количества клиентов в сети системы, а также их малой производительности.

1.1.2 Распространение на платформах

На данном этапе были изучены приложения, представленные в магазинах App Store и Google Play. Аналоги разрабатываемого приложения распространяются по схеме In-App Purchases [1]. Существуют несколько типов таких приложений, самыми распространенными из них являются следующие:

- пользователь, скачивая приложение из магазина, получает доступ к его полной версии, и оплата идет непосредственно за услуги, предоставляемые сервисом. Покупки внутри приложений представляют из себя подписки или единоразовые платежи, дающие определенные преимущества в использовании, не являющийся дополнительным функционалом приложения скидки на поездки в такси определенного класса;
- пользователь, скачивая приложение из магазина, получает доступ к его ограниченной версии, покупка (подписка или единоразовый платеж) разблокируют недоступный ранее платный функционал.

Сервисы, представленные в магазинах предложений, являются представителями первого типа приложений.

1.2 Обзор аналогов

1.2.1 Yandex.Go

Yandex.Go – это бесплатное приложение, ориентированное на рынок стран СНГ, для пользования услугами такси, перевозок, а так же доставки еды [2]. Грамотно продуманный и реализованный графический интерфейс, позволяющий выполнять все самые популярные задачи одной рукой в несколько нажатий, обеспечивает удобство пользования. Интерфейс приложения представлен на рисунке 1.2.

В Yandex.Go реализована возможность оплаты поездки прямо из приложения. Пользователи могут привязывать карты различных банков и использовать одну из них для последующей оплаты. При списании средств не требуется одноразовый пароль (OTP), это делает оплату моментальной и исключает вмешательство пользователя в процесс.

Важным при использовании таких приложений является точность расчета времени прибытия и показа местоположения. В приложении информация о загруженности дорог и, соответственно, основанный на этих данных маршрут и расчетное время прибытия, вычисляются исходя из данных других сервисов компании Yandex. Необходимые данные вычисляются с использованием сервисов Yandex.Maps и Yandex.Navigator, которые анализируют данные со спутников, подключенных к сети устройств и различных государственных источников, с которыми сотрудничает данная компания.

Бесплатная версия приложения предоставляет пользователю функционал, необходимый для заказа такси и перевозки вещей. При покупке подписки пользователь получает доступ к различным бонусам в системе компании Yandex, например:

- бесплатный доступ к различным сервисам компании, таким как Kinopoisk, Yandex.Music и т.д.;
- скидки на поездки в такси определенного класса.

Плюсы сервиса Yandex.Go:

- возможность оплаты картой;
- высокая точность расчета времени прибытия;
- удобный пользовательский интерфейс.

Минусы приложения:

- реализация приложения под каждую из платформ, что увеличивает команду разработки;
- отсутствие возможности оплаты мобильными системами;
- функционирует только в некоторых странах СНГ.

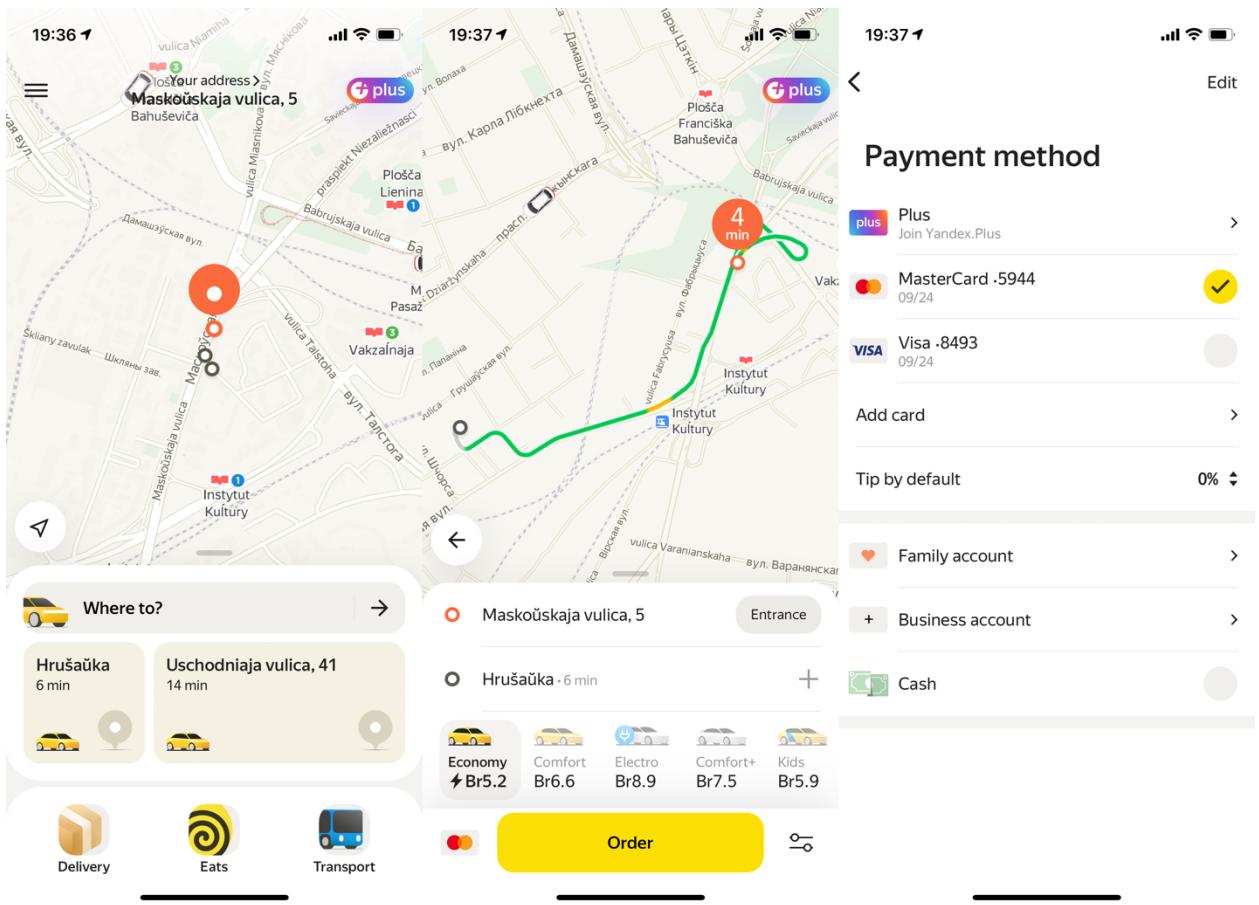


Рисунок 1.2 – Приложение Yandex.Go

1.2.2 Uber

Uber – это мобильное приложение для пользования услугами такси международного уровня: сервис является крупнейшим игроком на многих мировых рынках, в том числе американский, европейский и рынок стран СНГ [3]. Пользовательский интерфейс приложения представлен на рисунке 1.3. Бесплатная версия позволяет вызывать такси различных классов и заказывать доставку еды. Преимуществом приложения являются дополнительные системы оплаты.

В Uber, как и приведенном выше аналоге, пользователь может добавлять свои платежные карты в персональный аккаунт, однако помимо этого у пользователя появляется возможность оплаты платформенными сервисами, такими как Apple Pay, Google Pay и т.д., что позволяет пользователю избегать предоставления своих персональных данных третьим лицам.

Как и в большинстве аналогичных приложений, пользователь может получить доступ к дополнительному функционалу приложения с помощью подписки. Дополнительные функции, предоставляемые по подписке, связаны с получением различных скидок и бонусов. В бесплатной версии приложения вызов такси или доставки предполагает оплату по обычной цене с надбавкой в качестве платы за услуги сервиса. При покупке подписки у пользователя

появляется дополнительная скидка на такси и доставку, а также появляются дополнительные места, из которых можно заказывать еду.

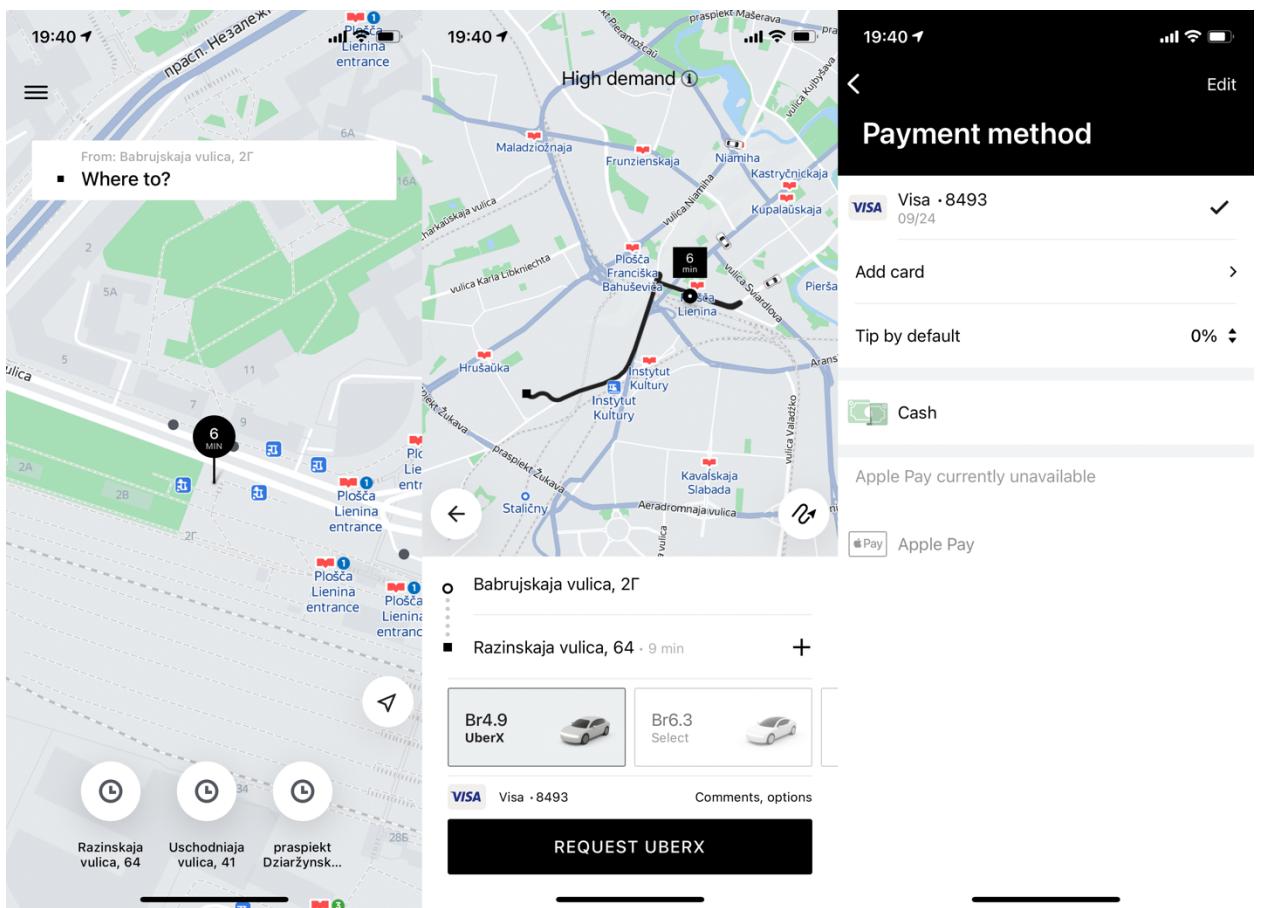


Рисунок 1.3 – Приложение Uber

Плюсы сервиса Uber:

- возможность оплаты картой и мобильными системами (Apple Pay, Google Pay);
- высокая точность расчета времени прибытия;
- работает в большинстве стран мира.

Минусы приложения:

- реализация приложения под каждую из платформ, что увеличивает команду разработки;
- недостатки в пользовательском интерфейсе (например расположение строки поиска сверху экрана, при использовании телефона одной рукой требуется перехватывать устройство).

1.2.3 Bolt

Bolt дает возможность заказывать такси в различных точках мира [4]. Пользовательский интерфейс данного приложения представлен на рисунке

1.4. Особенностью данного сервиса является экологичность транспорта и высокая квалификация водителей, что делает его лучшим на рынке Европы. Сервис предоставляет удобный интерфейс и высокую отзывчивость как приложения, так и его серверов, что делает работу с ним плавной и быстрой. Преимуществами данного приложения является возможность оплаты через платформенные сервисы, а также расширенные возможности по авторизации.

Bolt предоставляет пользователям возможность заказа такси, однако, в отличие от конкурентов, оно позволяет использовать уже существующие аккаунты различных сервисов и социальных сетей для авторизации в своей системе, что избавляет пользователя от необходимости предоставления данных и заполнения форм регистрации.

Bolt является абсолютно бесплатным и не предоставляет подписок для своих пользователей, зарабатывая исключительно на перевозках и транспортировке.

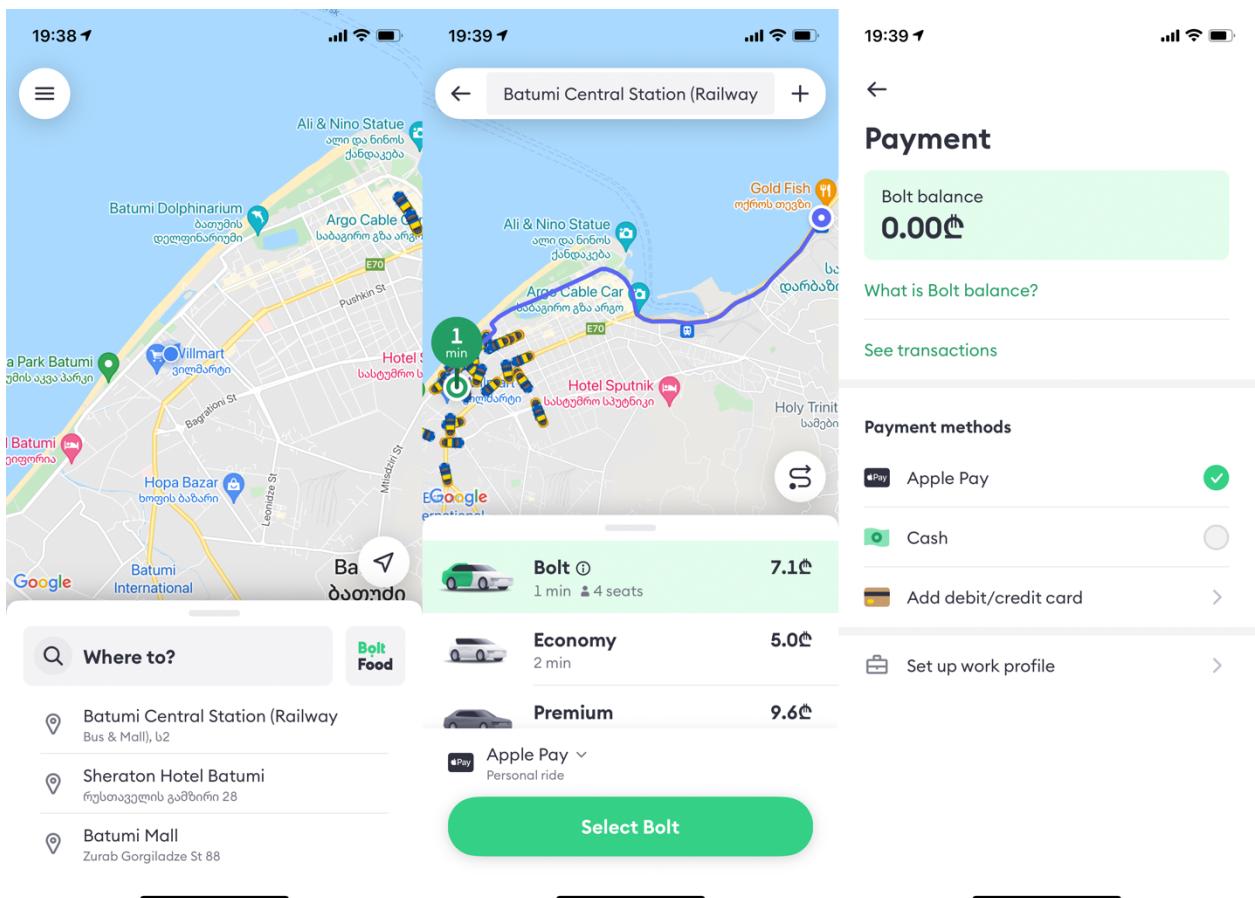


Рисунок 1.4 – Приложение Bolt

Плюсы сервиса Bolt:

- возможность оплаты картой и мобильными системами (Apple Pay, Google Pay);
- возможность авторизации с помощью сторонних сервисов;

- удобный пользовательский интерфейс и высокая скорость работы.

Минусы приложения:

- реализация приложения под каждую из платформ, что увеличивает команду разработки;
- списание средств происходит до подтверждения заказа водителем;
- функционирует в ограниченном количестве стран.

1.2.4 135

135 – сервис для пользования услугами такси и перевозками на территории Республики Беларусь [5]. Он является самым крупным исключительно белорусским сервисом, а также одним из первых на рынке. Данный сервис предоставляет для пользователей не только мобильное приложение, но еще и сайт, что позволяет охватить наибольший круг пользователей. Внешний вид мобильного приложения представлен на рисунке 1.5.

Приложение 135, как и Bolt, является абсолютно бесплатным и не предоставляет моделей подписки.

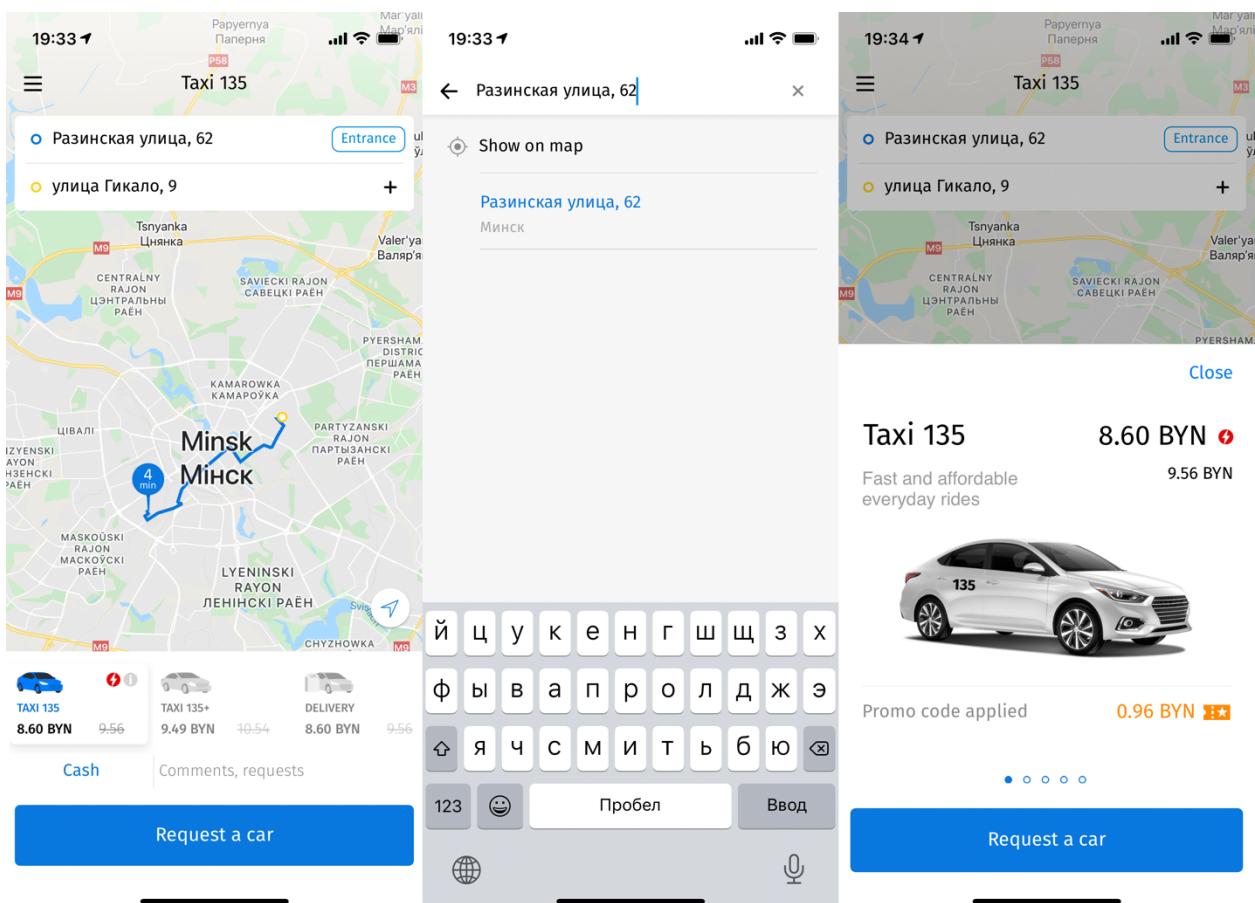


Рисунок 1.5 – Приложение 135

Плюсы сервиса 135:

- возможность оплаты картой;
- возможность вызова микроавтобусов при поездке большой компанией.

Минусы приложения:

- реализация приложения под каждую из платформ, что увеличивает команду разработки;
- малое количество машин;
- отсутствие возможности оплаты мобильными системами.

1.2.5 Maxim

Maxim – сервис по вызову такси, работающий в более чем 1000 городах мира [6]. Основанная в 2003 году, платформа начала набирать популярность в России, а позже начала распространяться в странах ближнего зарубежья. Пользовательский интерфейс представлен на рисунке 1.6. Несмотря на устаревший дизайн, приложение обладает достаточно широким функционалом, который отсутствует у аналогов.

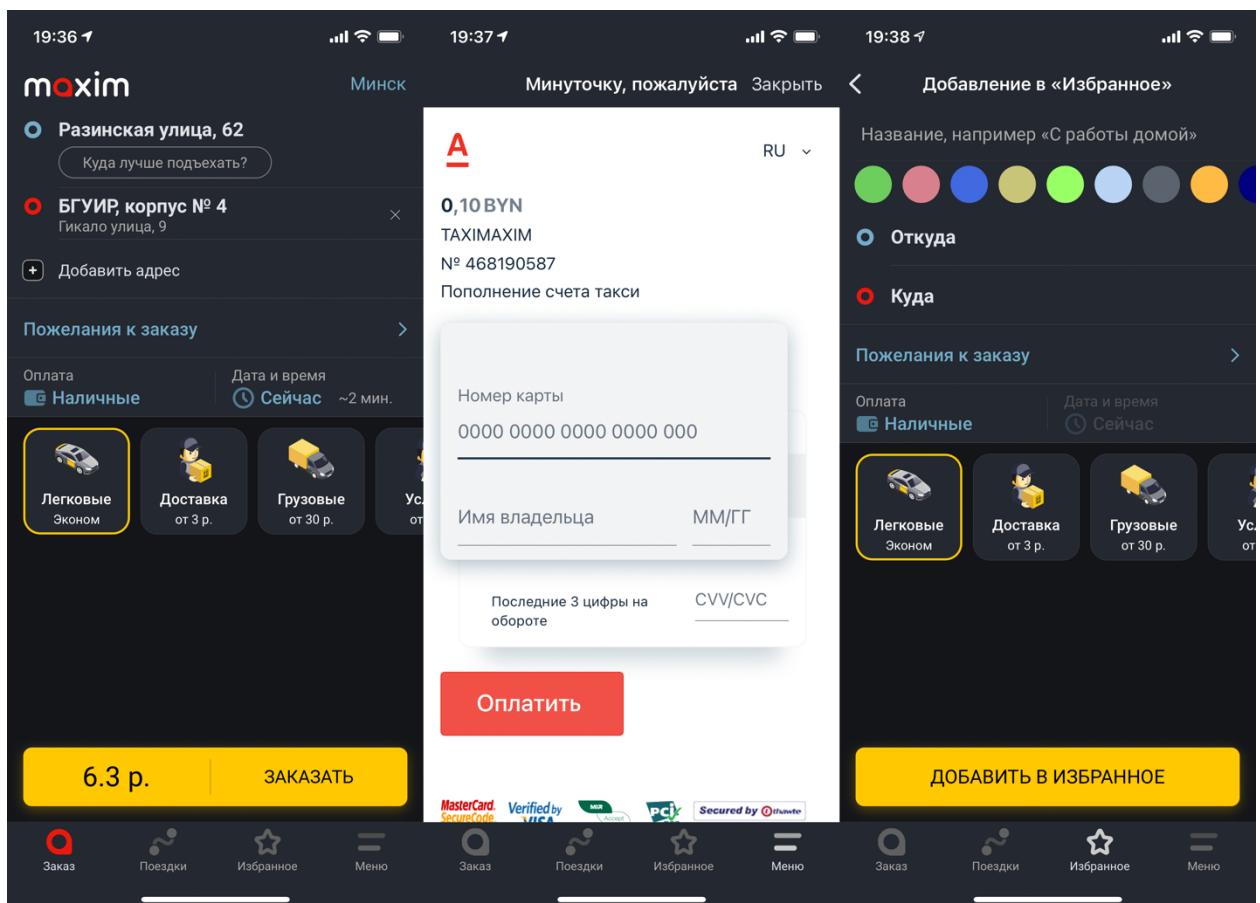


Рисунок 1.6 – Приложение Maxim

Приложение Maxim распространяется бесплатно и не предоставляет подписочную модель.

Плюсы сервиса Maxim:

- возможность оплаты картой;
- возможность добавление избранных маршрутов;
- возможность выбрать время подачи машины

Минусы приложения:

- добавление карты происходит в веб-интерфейсе;
- устаревший дизайн;
- малое количество машин.

1.3 Обзор технологий и инструментов

1.3.1 Клиентская часть

Клиентская часть разрабатываемой системой будет представлена мобильным приложением. Для выбора оптимальной технологии требуется сделать обзор доступных вариантов.

1.3.1.1 Android SDK

Android SDK – официальный набор утилит от Google для разработки под операционную систему Android [7]. Разработана с использованием языков Kotlin, Java и C++. Позволяет разрабатывать высокопроизводительные приложения для платформ Android, Android TV и WearOS.

1.3.1.2 iOS SDK

iOS SDK – официальный набор утилит от Apple для разработки под операционную систему iOS [8]. Разработана с использованием языков Objective-C, Swift и C++, позволяет разрабатывать оптимизированные приложения для модельного ряда устройств iPhone.

1.3.1.3 React Native

React Native – фреймворк для создания кроссплатформенных приложений, разработанный компанией Facebook [9]. Данный фреймворк позволяет разрабатывать приложение под несколько платформ (например, iOS и Android) с использованием лишь одного языка – TypeScript. Это преимущество компенсируется более низкой производительностью в сравнении с Android и iOS SDK.

1.3.1.4 Flutter

Flutter - фреймворк для создания кроссплатформенных приложений, разработанный компанией Google [10]. Как и React Native, данный фреймворк позволяет писать код на несколько платформ с использованием одного языка – Dart. Показатели производительности фреймворка также ниже, в сравнении с Android и iOS SDK.

1.3.2 Серверная часть

Серверное ПО является центром в клиент-серверной архитектуре: тут происходит обработка данных, платежей и прочая специфичная каждому приложению бизнес-логика. Рассмотрим самые популярные технологии для построения такого ПО.

1.3.2.1 Spring

Spring Framework — универсальный фреймворк с открытым исходным кодом для Java-платформы [11]. Spring дает большую свободу и гибкость разработчикам, при этом предоставляя эффективные и мощные утилиты для работы с такими важными вещами, как безопасность и хранение данных.

1.3.2.2 Nest

Nest (NestJS) — это фреймворк для создания эффективных масштабируемых серверных приложений NodeJS. Он сочетает в себе элементы ООП (объектно-ориентированное программирования), ФП (функционального программирования) и ФРП (функционально-реактивного программирования).

1.3.2.3 Django

Django — это высокоуровневый фреймворк, который способствует быстрой разработке и чистому прагматичному дизайну. Является самым популярным средством разработки веб-серверов на языке Python с широким набором встроенных средств и утилит.

1.3.3 Система управления базами данных

База данных является важной частью многопользовательских систем. При проектировании таких приложений выбор базы данных проводится путем строгого анализа и отбора по целому ряду пунктов, например скорость и отказоустойчивость. Для данного проекта оптимальными являются несколько СУБД, описанных ниже.

1.3.3.1 PostgreSQL

PostgreSQL — свободная объектно-реляционная система управления базами данных [13]. Существует в реализациях для множества операционных систем, в том числе Windows, macOS и Linux. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL 2011.

1.3.3.2 MongoDB

MongoDB — документно-ориентированная система управления базами данных, не требующая описания схемы таблиц. На данный момент является самой популярной NoSQL системой, в качестве схемы данных используется JSON.

1.4 Постановка задачи

Разрабатываемая система должна выдавать максимальную производительность при минимальных временных и денежных затратах на разработку.

Клиентская часть должна быть отзывчивой и не вызывать дискомфорт у пользователя, а также должно предоставлять следующие функции:

- функция регистрации и входа в систему;
- возможность добавления и оплаты картой;
- возможность выбора места отправки и назначения на карте или в поле для ввода адреса;
- просмотр истории поездок и профиля пользователя;
- отслеживание местоположения водителя и машины;
- принятие заказа, если пользователь приложения – водитель такси.

Данный функционал возможно реализовать с использованием любой из предоставленных выше технологий, однако ввиду необходимости удешевления и ускорения разработки, а также получения максимальной производительности, был выбран React Native из-за кроссплатформенности и достаточных показателей производительности.

Серверная часть должна максимально быстро обрабатывать входящие запросы клиентов и поддерживать с ними соединение для передачи текущего местоположения. Серверное ПО должно соответствовать следующим требованиям:

- высокая отказоустойчивость;
- высокие показатели по пропускной способности;
- эффективное расходование ресурсов;
- минимальное время, необходимое для разработки, внесения изменений и разворачивания системы.

Исходя из поставленных выше требований была выбрана микросервисная архитектура: логика сервера разбивается на несколько независимых, которые общаются между собой посредством некоторого протокола. Это было сделано для достижения высокой отказоустойчивости и увеличения пропускной способности. Задачи серверной части системы разбиваются на четыре основные части:

- получение данных о местоположении пользователей и водителей и поиск ближайшего водителя будет происходить в сервисе на отдельной машине. Данный сервис будет разработан на фреймворке Nest ввиду высокой пропускной способности NodeJS;
- данные пользователей будут обрабатываться в сервисе, написанном с использованием фреймворка Spring ввиду его высокой скорости и безопасности;
- обработка платежей будет происходить в сервисе, разработанном с использованием фреймворка Nest, так как множество платежных систем имеют свои утилиты для работы с языком TypeScript, на котором разработан Nest;
- расчет кратчайшего маршрута будет происходить в сервисе, написанном на Nest, так как Google предоставляет собственные удобные утилиты для работы с их картами и API для языка TypeScript.

Выбранные фреймворки позволяют эффективно использовать ресурсы и возможности системы, а также ускоряют работу по разворачиванию и разработке, благодаря чему выполняются все требования, поставленные к серверному ПО.

База данных в системе должна быть максимально быстрой и отказоустойчивой, поиск и запись в ней должны происходить за минимальный промежуток времени. Исходя из этих требований была выбрана реляционная база PostgreSQL, так как она является легкоразворачиваемой и бесплатной, а также позволяет производить быстрый и оптимизированный поиск по ключам таблиц.