# **Целевая аудитория**

## **1.1 Целевая аудитория**

Программное средство «Информационная система для просчёта стоимости маршрута каршеринга» позволяет клиентам просматривать не только ближайшие автомобили, но и просчитывать оптимальные тарифы.

В данный момент многие люди сталкиваются с проблемой просчёта оптимальных тарифов, т.к. операторов становится всё больше, а существующие на данный момент решения не полностью удовлетворяют запросы пользователей.

Примерная целевая аудитория: люди со средним и высоким достатком, с возрастным ограничением в восемнадцать лет.

## **Роли пользователей**

В приложении присутствуют три вида пользователей: администратор, гость и авторизованный пользователь. У каждого пользователя есть свои права, предназначенные для выполнения определённых операций.

Администратор имеет доступ к редактированию данных о тарифах, т.к. используемый сторонний веб-сервис оперирует не точными тарифами.

Гость имеет права только на просмотр карты автомобилей и просчёт тарифа.

Авторизованный пользователь имеет те же права что и гость, но имеет функциональность сохранения своих маршрутов.

# **Функционал ПО**

Информационная система имеет следующий функционал:

− управление базой автомобилей (добавлять и менять цены тарифов, добавлять цены тарифов для новых автомобилей): администратор;

* авторизация: администратор, зарегистрированный пользователь;
* изменение профилей пользователей: администратор;
* выполнение регистрацию: гость;
* просматривание карты расположения автомобилей: гость, зарегистрированный пользователь;
* получение расчёта тарифов по маршруту: гость, зарегистрированный пользователь;
* сохранение своих маршрутов: зарегистрированный пользователь;
* сохранение своих фильтры подбора автомобилей: зарегистрированный пользователь.

# **Системные требования**

## **3.1 Используемые технологии**

Проект реализован на основе существующего Android и iOS приложения carshare, которое является агрегатором машин минских машин каршеринга. В частности, будет использован работающий веб-сервис, для получения списка доступных автомобилей.

Необходимый дополнительный функционал: сервис для просчёта тарифов, сервис для работы с учётными записями пользователей и клиентское веб-приложение. Для реализации данного функционала используются следующие технологии: ASP.NET Core, Mongo DB – для серверной части, и React c Next.js – для клиентской. В качестве провайдера карт выбран 2ГИС.

ASP.NET Core – это новый общедоступный и кроссплатформенный фреймворк для создания современного облака приложений, связанных с подключением к интернету, таких как веб-приложения, приложения для интернета вещей и мобильных серверов. Приложения ASP.NET Core могут работать на .NET Core или на полной платформе .NET Framework. Этот фреймворк был спроектирован таким образом, чтобы обеспечить оптимизированную платформу разработки для приложений, которые перемещаются в облако или выполняются локально. Он состоит из модульных компонентов с минимальной перегрузкой, поэтому вы сохраняете гибкость при построении своих решений. Существует возможность разрабатывать и запускать кроссплатформенные ASP.NET Core приложения на Windows, Mac и Linux. Фреймворк ASP.NET Core общедоступен на GitHub.

MongoDB – система управления базами данных, которая работает с документоориентированной моделью данных. В отличие от реляционных СУБД, MongoDB не требуются таблицы, схемы или отдельный язык запросов. Информация хранится в виде документов либо коллекций.

React – это JavaScript библиотека GUI (англ) с открытым исходным кодом, сосредоточенная на одной конкретной цели – эффективном выполнение задач в рамках разработки пользовательского интерфейса. Его можно отнести к категории “V” в архитектурном шаблоне MVC (модель-вид-контроллер).

Next.js – это фреймворк, основанный на React, который позволяет создавать веб-приложения с улучшенной производительностью и улучшенным пользовательским опытом с помощью дополнительных функций предварительного рендеринга, таких как полноценный рендеринг на стороне сервера (SSR) и статическая генерация страниц (SSG).

## **3.2 Архитектура**

В основе будет использоваться микросервисная архитектура. Микросервисная архитектура – распространенный подход к разработке программного обеспечения, когда приложение разбивается на небольшие автономные компоненты (микросервисы) с четко определенными интерфейсами. Именно эта архитектура характерна для cloud-native приложений, которые сейчас популярны благодаря преимуществам, что открывают для бизнеса облачные среды.

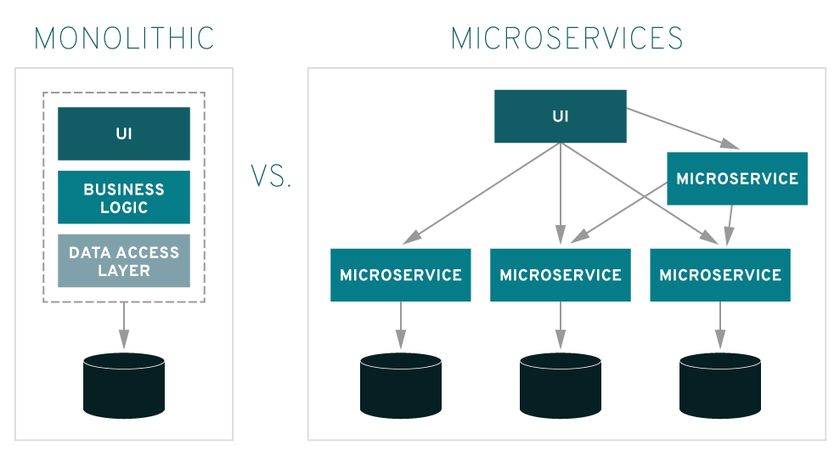


Рисунок 3.1 – Отличия архитектур

Микросервисное приложение состоит из множества мелких независимых и слабо связанных между собой сервисов, в то время как в монолите все его компоненты тесно взаимосвязаны и работают как единый сервис. Помимо прочего, это значит, что если какой-то один процесс в приложении с монолитной архитектурой становится более востребованным, приходится масштабировать всё приложение в целом. Сбой в каком-то одном процессе может поставить под угрозу всю систему. Наконец, такая сложность ограничивает возможности модернизации и затрудняет внедрение новых идей.

В информационной системе будут реализованы микросервисы для авторизации, пользовательских данных, расчёта маршрута и клиентского интерфейса.

## **3.3 Описание данных**

Для хранения пользовательских и конфигурационных данных используется Mongo DB. MongoDB – это документно-ориентированная база данных NoSQL, используемая для хранения больших объемов данных.

Документы состоят из пар ключ-значение, которые являются основной единицей данных в MongoDB. Коллекции содержат наборы документов и функции, которые эквивалентны таблицам реляционной базы данных. MongoDB – это база данных, появившаяся в середине 2000-х годов.

Ниже приведены структуры объектов для хранения данных информационной системы.

Поля объектов User:

− id, идентификатор пользователя;

­− userName, имя пользователя;

* hashedPassword, захешированный пароль;
* securityStamp, штамп для проверки изменения данных авторизации;
* userRole, роль пользователя.

Поля объектов UserRoute:

− id, идентификатор маршрута;

­− routePoints, массив координат маршрута;

* routeName, имя маршрута.

Поля объектов Tarrif:

− id, идентификатор тарифа;

­− operatorName, название оператора;

* carName, марка машины;
* prices, объект (список) цен на данную марку машины.

На рисунке 3.2 показаны коллекции этих объектов в БД.

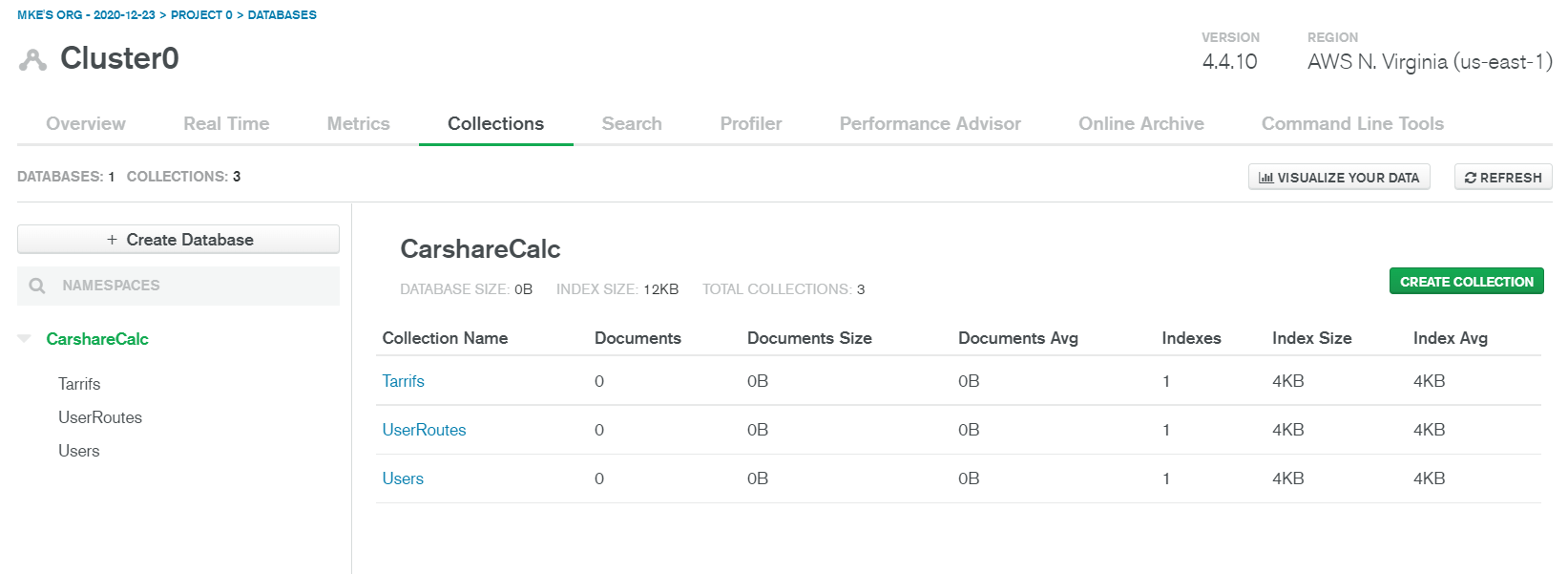


Рисунок 3.2 – Коллекции БД

# **Диаграммы**

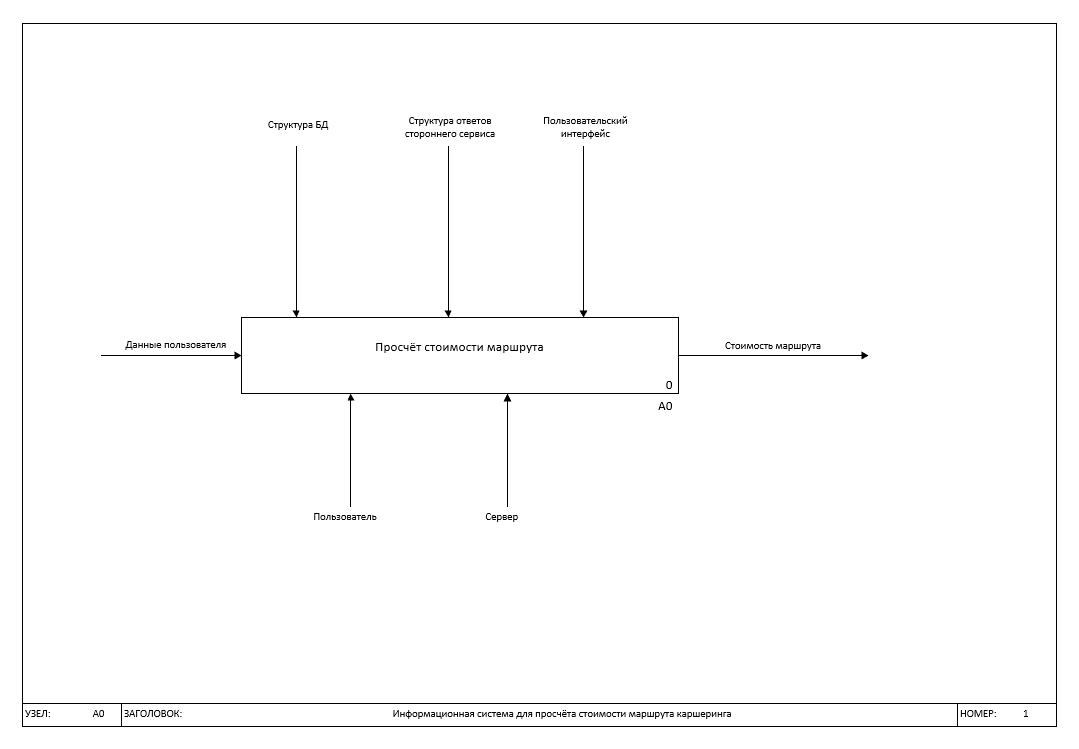


Рисунок 4.1 – Контекстная диаграмма

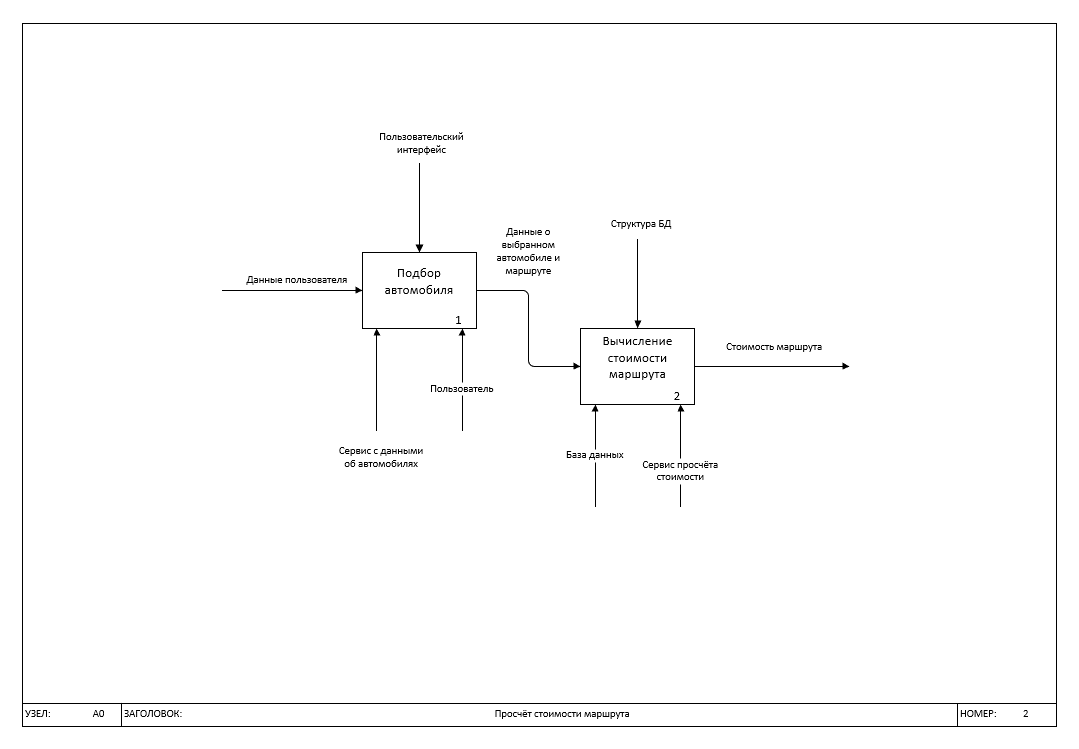


Рисунок 4.2 – Диаграмма первого уровня декомпозиции