Санкт-Петербургский государственный университет

Зауди Яссин

Выпускная квалификационная работа

Разработка веб-сервера для приложения (аренда самокатов) в Марокко

Уровень образования: бакалавриат

Направление 02.03.03 “Математическое обеспечение и администрирование информационных систем”

Основная образовательная программа СВ.5006.2019 “Математическое обеспечение и администрирование информационных систем”

Научный руководитель:
д.ф.-м.н, профессор кафедры информатики Т.М. Косовская

Рецензент:
к.ф.-м.н, доцент Кафедры системного программирования Д.В. Луцив

Санкт-Петербург
2023

Saint Petersburg State University

Zaoudi Yassine

Bachelor's Thesis

Development of a web server for an application (scooter rental) in Morocco

Education level: bachelor

Speciality 02.03.03 "Software and Administration of Information Systems"

Programme CB.5006.2019 "Software and Administration of Information Systems"

Scientific supervisor:

M.D. of Sc., Kosovskaya T.M.

Reviewer:
PhD Dmitry Luciv

Оглавление

1. Введение

Мобильность, практичность и экологичность стали важнейшими качествами XXI века. Они ценятся не только в человеке, но и в окружающих его предметах. Люди выбирают скорость, простоту, комфорт. Этим объясняется нарастающий тренд на ультрасовременные гаджеты, такие как электросамокаты.

Основной задачей и темой данного диплома стала разработка веб-сервера для приложения кикшеринговой компании.

Приложение аренды самокатов представляет из себя пользовательский интерфейс и логику, скрытую за ним. Эта логика и реализуется посредством веб-сервера. Веб-сервер – это “мозг” приложения, на сервере выполняются все взаимодействие между самокатами, пользователем и внешними сервисами, именно поэтому реализация данной важнейшей части приложения стала темой этой дипломной работы.

Подобные сервисы уже существуют по всему миру, в том числе и в России. Наибольшую долю рынка занимают сервисы от Яндекса и Whoosh. Эти сервисы представляют из себя сложные приложения с точки зрения разработки, и подобные решения не представлены в открытом доступе, поэтому в данной работе все было реализовано с нуля.

При таком подходе возникает множество вопросов и проблем с реализацией того или иного функционала, что и формирует задачи проекта. Разрабатывая большое приложение с нуля, нужно учесть много факторов и нюансов, чтобы финальный продукт работал стабильно, эффективно и решал поставленную задачу.

Так как исходный код существующих сервисов не представлен в открытом доступе, важно обратить внимание на устройство этих сервисов хотя бы со стороны пользователя и использовать удачные решения и подходы, а также учесть чужие недостатки и стремиться сделать свой сервис лучше.

Марокко - одно из самых развитых государств Африки. Я гражданин этой страны, а сервис краткосрочной аренды электросамокатов - мой стартап. Запуск проекта запланирован в Касабланке, в самом крупном городе королевства.

Для жителей Касабланки самокаты станут неотъемлемой частью повседневной жизни, так как инфраструктура города и погодные условия позволяют пользоваться ими круглый год. В то же время, в стране слабо развиты технологии и запуск такого масштабного проекта определенно ускорит ее развитие.

Именно поэтому в данной работе были приложены все усилия, чтобы создать правильный с точки зрения разработки продукт, а именно, его главную функциональную часть, отвечающую за стабильную и быструю работу – веб-сервер приложения.

2. Постановка задач

Целью работы является создание веб-сервера для приложения аренды самокатов, позволяющего пользователям легко находить и бронировать свободные самокаты в нужном месте и время.

Для достижения поставленной цели были сформулированы следующие задачи.

- Произвести обзор аналогичных сервисов.
- Разработать требования к веб-серверу.
- Разработать архитектуру веб-сервера.
- Реализовать работу веб-сервера.

3. Обзор

Технологии, обеспечивающие серверное взаимодействие и работу подобных сервисов развиваются очень быстро и перспективно. Важно постоянно следить за трендами и обновлениями в данной сфере и обращать внимание на методы, используемые лидерами в отрасли.

Электросамокаты – не исключение. Технологии для создания функционала такого сервиса постоянно развиваются, а сервисы аренды становятся все популярнее среди жителей городов во всем мире.

Данный вид транспорта распространен и в России. Популяризаторами и самими крупными представителями отрасли в России являются компании Яндекс [1], Whoosh [2] и Юрент [3]. Каждая из них имеет свою собственную технологию и мобильное приложение для аренды самокатов.

В данном обзоре представлен анализ аналогичных сервисов, которые уже существуют на рынке аренды самокатов. Были рассмотрены сервисы от Whoosh, Яндекс, Юрент и Lime [4], проанализированы их основные особенности и преимущества, а также выявлены недостатки и возможности для улучшения. Это сделано с целью лучше понять, какие функции и возможности необходимо реализовать, чтобы обеспечить максимально стабильную работу и лучший пользовательский опыт. Кроме того, будет рассмотрена актуальность выбранного решения.

3.1 Анализ аналогичных сервисов.

С целью создать стабильно работающий и надежный сервис было принято решение проанализировать уже существующие на рынке решения, попользоваться ими на практике, выявить их достоинства и недостатки и опираться на эту информацию при дальнейшей работе.

В крупнейших городах России самыми популярными сервисами аренды самокатов являются Whoosh [2], Яндекс [1] и Юрент. На рынке США и Европы самым крупным сервисом является Lime [4].

3.1.1 Whoosh

Данный стартап привлек внушительные инвестиции от Сколково и был первым крупным проектом из данной отрасли в России [5]. На данный момент эти самокаты можно арендовать практически во всех крупных городах. У данного сервиса были выявлены следующие недостатки и особенности:

К плюсам можно отнести следующие особенности:

- широкий выбор мест для аренды,
- большое количество самокатов,
- доступность во многих городах.

Главными недостатками является фактор того, что пользователи жалуются на неисправность самокатов.

При разработке функционала любого приложения полезно изучить готовые решения с целью упростить выбор архитектурных решений и технологий. К сожалению, невозможно посмотреть и изучить архитектуру приложения Whoosh, поэтому был проведен анализ функционала приложения от лица пользователя.

Приложение использует метод аутентификации и авторизации с использованием одноразовых кодов. Аналогичный метод был использован и в этой работе.

Также можно отметить, что приложение отображает доступные самокаты на карте, а так же зоны, в которых разрешена парковка. За данный функционал отвечает сервис отображения доступных самокатов.

В ходе пользования сервисом были обнаружены проблемы с работой веб-сервера приложения, а именно, аренда не всегда завершалась в нужной локации, что говорит о проблемах в работе сервисов контроля состояния самокатов и контроля заказов, а также о возможности нестабильной работы приложения.

3.1.2 Сервис аренды от Яндекса

Вслед за Whoosh крупнейшая российская IT-компания представила свой сервис аренды самокатов. О его преимуществах говорят следующие возможности.

- Возможность завершения аренды в практически любой зоне использования.
- Наличие самокатов собственной разработки.

А недостатками сервиса являются следующие особенности.

- Многочисленные жалобы пользователей на мобильное приложение.
- Небольшое количество самокатов (в сравнении с Whoosh).

Яндекс является одной из крупнейших IT-компаний в стране и имеет огромную экосистему сервисов и приложений. Одним из таких приложений является их сервис заказа такси, в который и была интегрирована аренда самокатов.

Так как аренда самокатов является одной из частей приложения Яндекс Go, найти функционал связанный с самокатами оказалось не просто. Приложение кажется перегруженным и не очень удобным для использования. Как выяснилось в ходе исследования, аналогичное мнение сложилось у многих пользователей сервиса.

В самом интерфейсе приложения находится карта, на которой изображены самокаты и зоны парковки, аналогично с Whoosh. В процессе пользования было обнаружено, что самокаты не всегда отображаются на карте корректно: неверное местоположение или отсутствие самокатов в указанных точках. Все это говорит о сбоях в работе сервисов контроля самокатов и отображения свободных самокатов. При работе над аналогичным сервисом в этой работе, были изучены актуальные методы и применены технологии, которые должны способствовать более стабильной работе сервисов.

3.1.3 Юрент

В 2018 году на юге России была основана кикшеринговая компания Юрент. Первое время компания предоставляла свои услуги только в

южных курортных городах России, но сейчас уже распространилась по всей стране. О преимуществах компании можно назвать следующие факты:

- широкое распространение на юге России,
- сравнительно небольшая стоимость аренды самокатов.

Говоря о недостатках, можно назвать следующие факты:

- не самые современные самокаты,
- плохая техническая поддержка самокатов,
- небольшое количество самокатов в крупных городах.

После использования сервиса можно заметить, что сервис заметно уступает конкурентам по части технической поддержки. Что касается функционала приложения, самокат можно забронировать, арендовать и завершить аренду. Также самокаты отображаются на карте в реальном времени при перемещении, аналогично с вышеописанными сервисами. Возникали проблемы во время использования сервиса, такие как некорренткое поведение самокатов во время поездки, а именно, неожиданное выключение самокатов, долгое начало и окончание аренды.

3.1.4 Lime

Данный стартап быстро набрал популярность и привлек внушительные инвестиции на рынке США. На данный момент эти самокаты можно арендовать практически во всех крупных странах. У данного сервисы есть следующие плюсы и минусы:

К плюсам можно отнести следующие наблюдения:

- широкий выбор мест для аренды,
- большое количество самокатов,
- доступность во многих городах.

Главными недостатками являются следующие факторы:

- высокие цены на аренду в высоконагруженных районах,
- пользователи жалуются на неисправность самокатов.

Данный сервис не представлен на территории России, поэтому выводы о функционале приложения были сделаны на основе информации в

интернете. Приложение имеет схожий с российскими сервисами функционал: пользователь может авторизоваться, посмотреть расположение самокатов, забронировать самокат и начать и закончить аренду.

3.1.3 Выводы

Подводя итог, можно сказать, что основными достоинствами крупных компаний для аренды самокатов является большая покрываемая территория для аренды, большое количество самокатов и, конечно же, мобильное приложение для аренды, разработке которого и посвящена данная работа.

	Начало аренды	Окончание аренды	Отслеживание самокатов	Отображение самокатов	Аутентификация
Яндекс	–	+/–	+	–	+
Whoosh	+	–	–	+	+
Юрент	–	–	+	+	+
Lime	+	+	+	+	+

Таблица 1: Сравнение функций сервисов по аренде самокатов.

На основе обзора этих сервисов были сформулированы требования к сервису, а также выделены главные недостатки работающих сервисов, а именно, серверной части. Это будет учтено в процессе реализации собственного продукта на практике.

3.2 Актуальность решения

Спрос на аренду самокатов в последнее время значительно возрос, особенно в крупных городах. Это связано с тем, что аренда самокатов является удобным и экологически чистым способом передвижения по городу.

Многие люди предпочитают арендовать самокаты вместо использования общественного транспорта или личного автомобиля, так как это позволяет сэкономить время и деньги на транспортировку. Кроме того, аренда самокатов позволяет избежать пробок и проблем с парковкой, что делает этот вид транспорта особенно привлекательным для жителей крупных городов.

Также аренда самокатов становится все более популярной среди туристов, которые хотят быстро и удобно осмотреть достопримечательности города. Аренда самоката позволяет им свободно передвигаться по городу и наслаждаться его красотами, независимо от общественного транспорта или экскурсионных автобусов.

В связи с растущим спросом на аренду самокатов, многие компании начали предлагать эту услугу, что привело к появлению новых игроков на рынке и снижению цен на аренду. Кроме того, некоторые города начали развивать инфраструктуру для самокатов, такие как специальные дорожки и парковки, что делает этот вид транспорта еще более привлекательным для пользователей.

Таким образом, спрос на аренду самокатов растет, и это отражает изменение потребительских предпочтений в пользу более удобных и экологически чистых способов передвижения.

4. Требования

В данной главе описаны основные требования к веб-серверу, которые были разработаны исходя из основных потребностей пользователей, функциональных возможностей аналогов, а также основной бизнес-логики приложения.

4.1. Функциональные требования

4.1.1. Аутентификация пользователей

Аутентификация – проверка подлинности данных, предоставленных пользователем.

В данном проекте аутентификация является неотъемлемой частью процесса проверки подлинности идентификации пользователя, который осуществляет запросы к системе. Кроме того, в контексте аренды самокатов, оплата производится с использованием карты пользователя, что подчеркивает необходимость обеспечения надежности и безопасности процесса аутентификации для предотвращения возможных попыток мошенничества и злоупотребления чужими данными.

4.1.2. Предоставление списка свободных самокатов

Для начала процесса аренды пользователи должны иметь возможность получения информации о наличии и местонахождении свободных самокатов. Для этого необходимо обеспечить доступ к геопозиционным данным, которые позволят определить местоположение свободных самокатов.

4.1.3. Добавление новых самокатов

Не всегда количество самокатов будет удовлетворять бизнес нуждам компании и спросу пользователей, иногда будет необходимо внести новый самокат в систему, для этого и нужна опция добавления новых самокатов.

4.1.4. Начало и окончание аренды

В системе сервиса будут одновременно работать десятки самокатов, и пользователи будут пользоваться ими регулярно. Поэтому необходима опция начала аренды и ее окончания, реализованная так, чтобы пользователь максимально просто и быстро арендовал самокат и завершал

аренду, а также сервер выдерживал все нагрузки, связанные с большим количеством самокатов и пользователей.

4.1.5. Сохранение геопозиции самокатов

Поскольку самокат является подвижным объектом, который может находиться в движении и подвергаться риску кражи, необходимо осуществлять постоянный мониторинг местоположения и уровня заряда самоката с небольшим временным интервалом. Это позволит оперативно реагировать на чрезвычайные ситуации и принимать меры по их предотвращению.

4.2. Нефункциональные требования

4.2.1. Модульность

Свойство системы, характеризующееся ее способностью разбиваться на более мелкие и независимые подсистемы (модули), каждый из которых описывает одну функциональную часть системы и может быть изменен или заменен без влияния на другие модули называется модульностью.

Для эффективной разработки и проектирования современного сервиса крайне важно обеспечить его модульность. Она позволит значительно упростить процесс доработки и тестирования сервиса, а также повысить его гибкость и масштабируемость.

4.2.2. Масштабируемость

Способность системы увеличивать или уменьшать свою емкость или производительность в зависимости от изменения объема входных данных или нагрузки на систему называется масштабируемостью. Это означает, что система должна справляться с ростом количества пользователей, увеличением объема данных или увеличением числа транзакций, сохраняя при этом высокую производительность и надежность.

4.2.3. Отказоустойчивость

Бизнес-процессы могут оказать влияние на потребности пользователей, что может привести к увеличению нагрузки на веб-сервер. Поэтому крайне важно обеспечить корректную работу сервера даже при повышенном спросе.

5. Архитектура

В данной главе описано построение архитектуры веб-сервера, учитывая функциональные и нефункциональные требования, а также описаны сервисы и их ответственности, приведены основные сценарии использования.

5.1. Построение архитектуры

При выборе архитектурного стиля было необходимо опираться на нефункциональные требования. Эти требования повлекли выбор микросервисного архитектурного стиля из-за его более гибкого масштабирования и улучшения отказоустойчивости, а также высокой модульности.

5.1.1. Разделение сервера на микросервисы

При разработке приложение разделяется на более мелкие сервисы, каждый из которых отвечает за свою конкретную функциональность. Например, один сервис может быть ответственным за управление свободными самокатами, другой за процесс оплаты, третий за авторизацию пользователей. Более того, так как используется микросервисная архитектура, данные сервисы абсолютно автономны и работают независимо друг от друга.

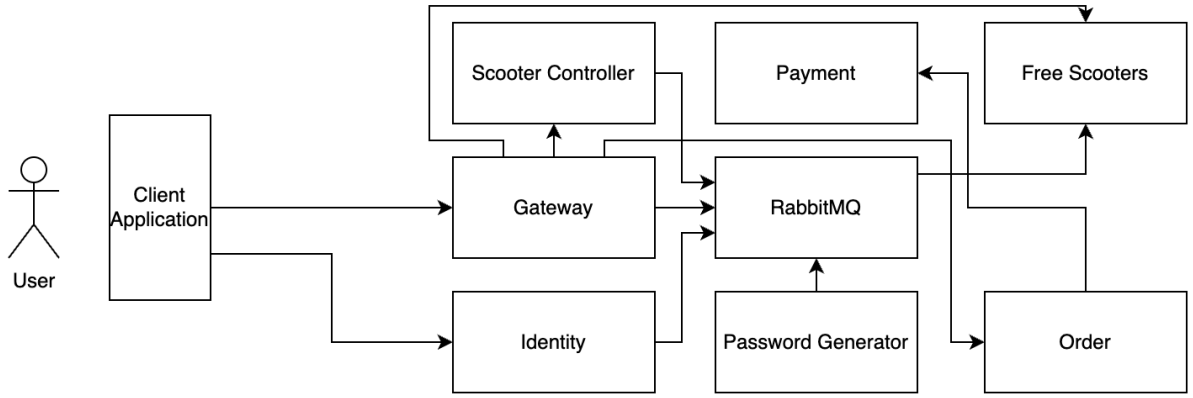

Рис. 1: Разделение веб-сервера на микросервисы

5.1.2. Gateway

Gateway – сервис-прослойка между клиентом и сервером, который управляет запросами и передает их нужным сервисам. Он предоставляет

единую точку доступа для всех клиентов и облегчает управление маршрутизацией запросов к различным микросервисам.

В данном проекте был реализован Gateway, который выполняет функции авторизации пользователей, а также маршрутизации запросов к сервисам.

5.1.3. Сервис аутентификации и авторизации пользователей

Для удобства пользователей при аутентификации и авторизации было решено использовать одноразовые пароли, которые высылаются на телефон пользователя через СМС-код или последние 4 цифры звонящего номера.

Был реализован сервис аутентификации и авторизации с использованием библиотеки Openiddict [6], которая лучше подходит для текущей аутентификации, так как позволила на более низком уровне реализовать проверку на совпадение логина и пароля.

Все данные пользователей хранятся в базе данных PostgreSQL [7], с использованием Entity Framework Core [8].

5.1.4. Сервис генерации паролей

Для генерации одноразовых паролей реализован отдельный сервис, который может как генерировать сам пароль, так и отправлять запросы на отдельные внешние сервисы отправки СМС.

В качестве стороннего сервиса выступает сервис, предоставляющий базу телефонных номеров, с которых пользователю могут поступать звонки. В качестве пароля выступают последние 4 цифры номера, с которого пользователь получил звонок.

Таким образом, когда сервис генерации паролей получает запрос, он посылает запрос на внешний сервис. Внешний сервис осуществляет звонок пользователю и возвращает 4 последние цифры использованного номера. Сервис генерации паролей отсылает пароль и номер телефона пользователя на сервис аутентификации.

5.1.5. Сервис для контроля состояния самокатов

Основная задача данного сервиса хранить и обрабатывать информацию о всех самокатах, а именно: уровне заряда, геопозиции, доступности для заказа.

Также сервис меняет состояния самоката, например, включает или выключать его.

Важной функциональной особенностью данного сервиса является возможность общения с сервисом, предоставляющим пользователям информацию о доступных для заказа самокатах.

5.1.6. Сервис отображения доступных самокатов

Сервис для отображения свободных самокатов включает в себя следующие функции:

- отображение данных о свободных самокатах,
- уведомление об изменениях в расположении самокатов,
- общение с сервисом контроля самокатов для получения актуальной информации о свободных самокатах.

Сервис предоставляет следующие данные о свободных самокатах:

- широта и долгота местоположения,
- название,
- идентификационный номер,
- уровень заряда.

5.1.7. Сервис контроля заказов

Для обеспечения корректной работы бизнес-процессов, связанных с созданием и управлением заказами, был разработан специальный сервис контроля заказов. Данный сервис позволяет создавать новые заказы, а также сохранять их состояния в базу данных. Он реализует необходимую логику, которая обеспечивает корректную работу бизнес-процессов и позволяет управлять заказами эффективно и надежно.

5.1.8. RabbitMQ

Для асинхронного общения между сервисами был использован брокер сообщений с открытым исходным кодом RabbitMQ [9]. Брокер сообщений был помещен в Docker-контейнер [10] и использовался как внутренний сервис.

5.2 Основные сценарии использования

В данном разделе описаны основные сценарии использования веб-сервера, а также диаграммы последовательностей, описывающие взаимодействия между сервисами.

5.2.1. Авторизация по одноразовому паролю

Авторизация по одноразовому паролю включает в себя следующие этапы:

1. Пользователь вводит свой номер телефона в клиентском приложении.
2. Клиентское приложение отправляет запрос на Gateway с просьбой сгенерировать одноразовый пароль по данному номеру телефона.
3. Gateway проверяет частоту поступления запросов от отправителя с данного IP-адреса. Если последний запрос был отправлен более минуты назад, то Gateway посылает сообщение на сервис по генерации паролей через асинхронное общение (сервис RabbitMQ).
4. Сервис генерации паролей получает запрос и генерирует новый код авторизации.
5. Сервис генерации паролей отправляет запрос на внешний сервис для отправки СМС с кодом авторизации на данный телефон, а также оповещает сервис аутентификации о новом пароле.
6. Пользователь вводит полученный код авторизации в клиентском приложении.
7. Клиентское приложение отправляет номер телефона и код на сервер аутентификации.

8. Сервер аутентификации проверяет правильность введенного кода и, в случае успеха, выдает токен доступа (access token) в ответ на запрос клиентского приложения.

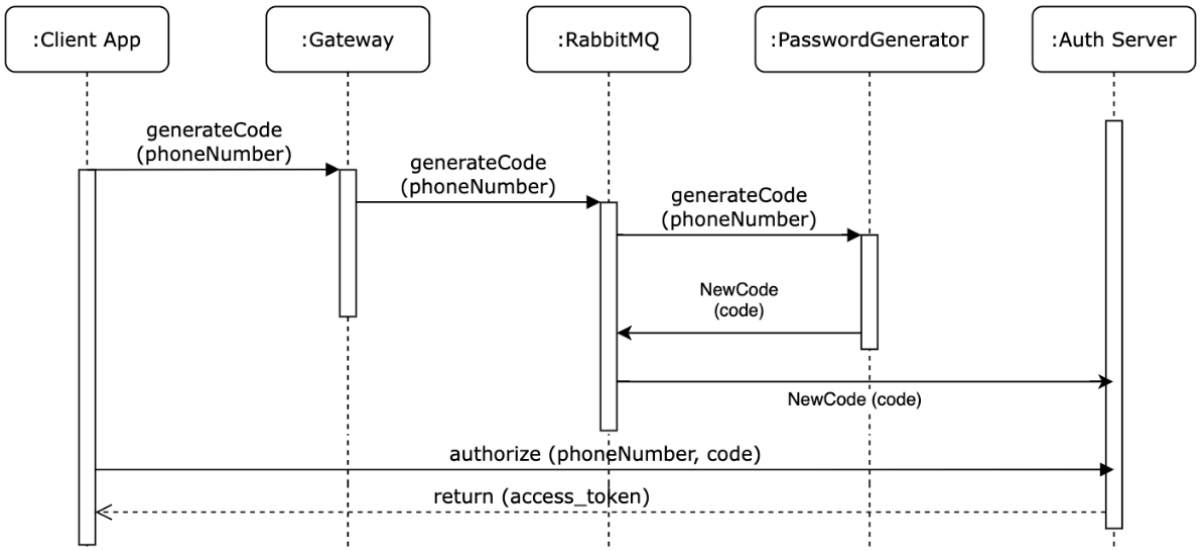

Рис. 2: Диаграмма последовательностей авторизации по одноразовому паролю

Важно отметить, что данный механизм авторизации является частью системы безопасности и обеспечивает защиту персональных данных пользователей. Он также позволяет удобно и быстро осуществлять авторизацию в приложении.

5.2.2. Добавление новых самокатов

Процесс добавления нового самоката включает в себя следующие этапы:

1. Администратор вводит информацию о новом самокате в клиентском приложении.
2. Клиентское приложение отправляет запрос на Gateway вместе с токеном доступа.
3. Gateway проверяет наличие разрешения "addNewScooter" и отправляет запрос на добавление самоката в Scooter-Controller.
4. Scooter-Controller добавляет новый самокат и возвращает идентификационный номер нового самоката.

5. Scooter-Controller отправляет асинхронное сообщение Free-Scooter через RabbitMQ, уведомляя о новом самокате.

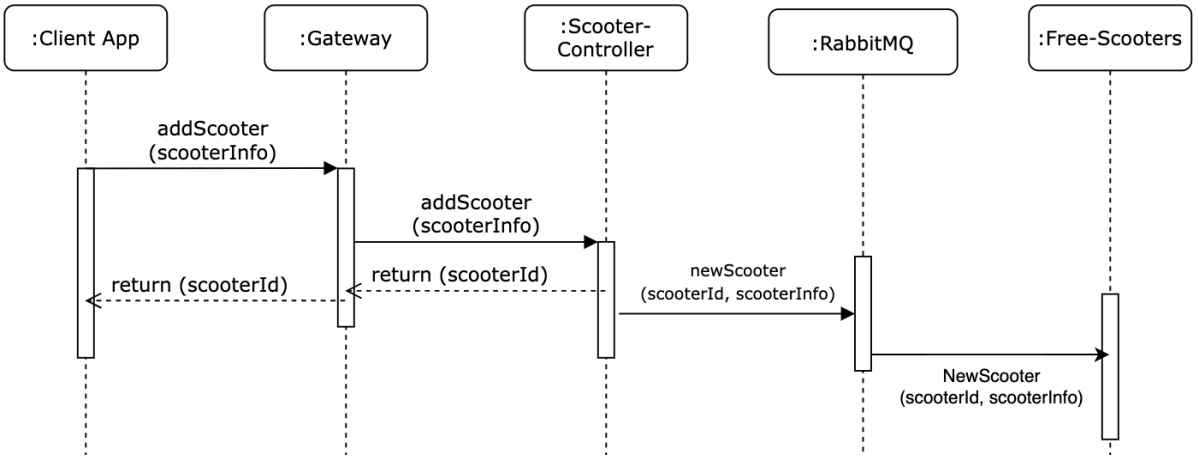

Рис. 3: Диаграмма последовательностей добавления нового самоката в систему

5.2.3. Зарядка самокатов

Для постановки самокатов на зарядку необходимо скрыть их от пользователей.

Это достигается следующим образом:

1. Менеджер по самокатам выбирает разряженный самокат в клиентском приложении и подтверждает его постановку на зарядку.
2. Клиентское приложение отправляет запрос на Gateway для начала зарядки.
3. Gateway отправляет запрос на Scooter-Controller с запросом на скрытие самоката от пользователей.
4. Scooter-Controller отправляет сообщение о скрытии самоката асинхронным способом на Free-Scooter с помощью RabbitMQ.
5. Free-Scooter обновляет текущее состояние самоката на "Скрыт".

Следует отметить, что самокат продолжает отправлять уведомления на Scooter-Controller для оповещения о своем состоянии, таких как геопозиция и процент заряда.

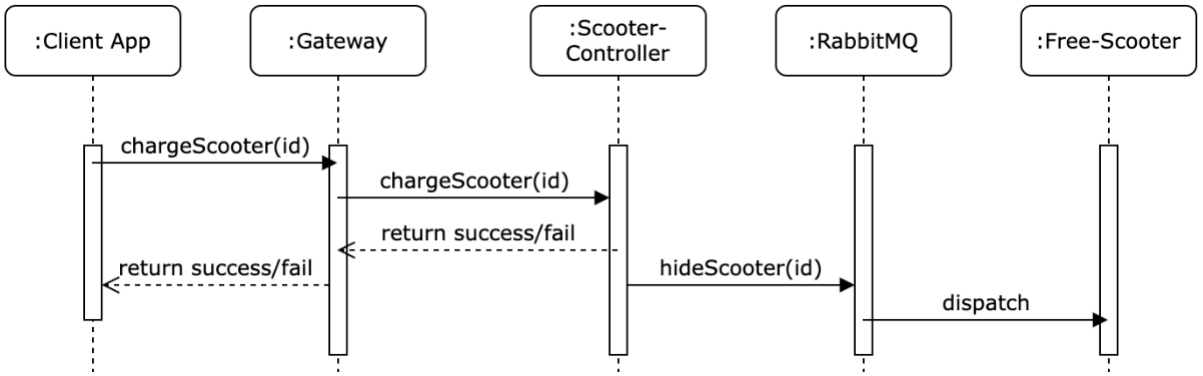

Рис. 4: Диаграмма последовательностей зарядки самокатов

Когда самокат заряжен и готов к использованию, происходит аналогичный сценарий, но Scooter-Controller отправляет сообщение Free-Scooter о показе самоката для пользователей.

5.2.4. Начало и окончание аренды самоката

Для начала аренды самоката пользователь должен узнать, какие самокаты доступны. Это достигается следующим образом:

1. Клиентское приложение отправляет запрос на Gateway для получения списка свободных самокатов.
2. Gateway перенаправляет запрос на Free-Scooter.
3. Free-Scooter возвращает список свободных самокатов с информацией о каждом, такой как геопозиция, процент заряда, идентификационный номер и название модели.

Процесс начала аренды самоката включает в себя следующие этапы:

1. Клиентское приложение отправляет запрос на Gateway о начале аренды с номером самоката.
2. Gateway извлекает идентификационный номер пользователя из токена доступа, прикрепляет его к номеру самоката и перенаправляет запрос на сервис контроля заказов (Order).
3. Сервис контроля заказов отправляет запрос на Scooter-Controller, чтобы убедиться, что самокат можно забронировать, то есть он не занят в данный момент времени. Затем сервис бронирует самокат за пользователем и отправляет запрос на сервис оплаты для списания или заморозки денежных средств на банковском счету пользователя.

4. Если все прошло успешно, сервис контроля заказов отправляет запрос на Scooter-Controller о готовности к поездке. Затем сервис регистрирует начало поездки и возвращает номер заказа (поездки).

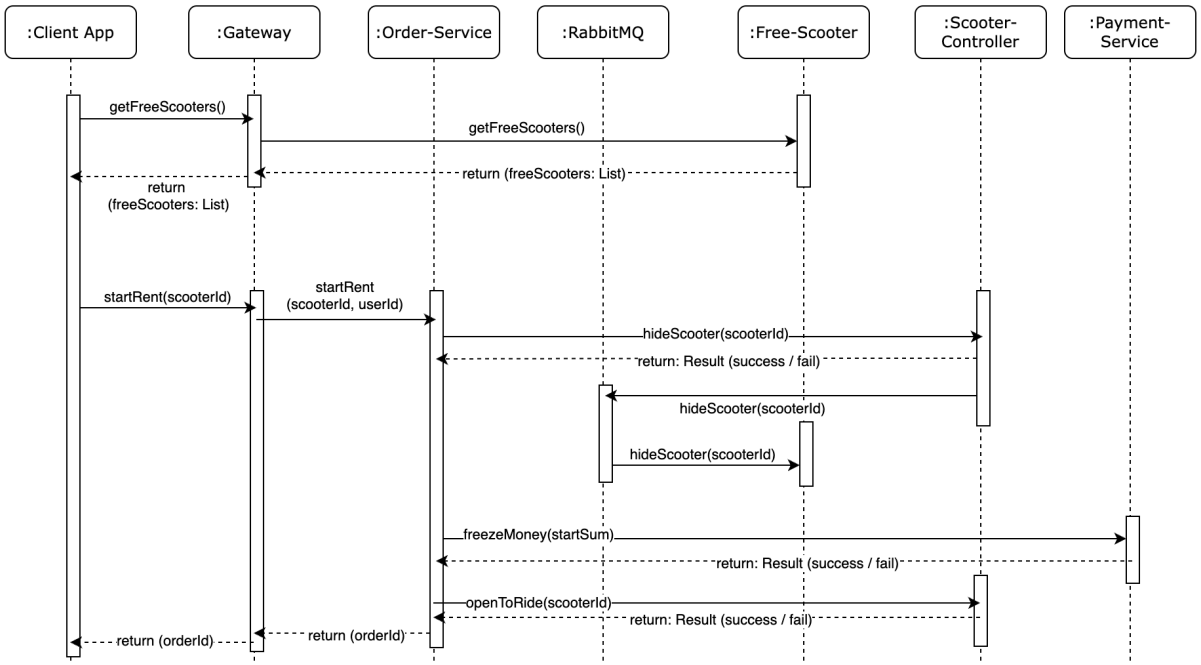

Рис. 5: Диаграмма последовательностей: получение списка свободных самокатов и начало аренды

Процесс окончания аренды самоката включает себя следующее:

1. Клиентское приложение отправляет запрос на Gateway об окончании аренды с номером самоката.

2. Gateway извлекает идентификационный номер пользователя из токена доступа, прикрепляет его к номеру самоката и перенаправляет запрос на сервис контроля заказов (Order).

3. Сервис контроля самокатов отправляет запрос на Scooter-Controller, чтобы убедиться, что аренду можно завершить, то есть самокат находится на парковке. Затем сервис по контролю за самокатами блокирует самокат и посылает асинхронное сообщение на Free Scooter о том, что самокат свободен к аренде.

4. Сервис по контролю за заказами (Order) рассчитывает стоимость поездки и отправляет запрос на сервис оплаты для списания конечной суммы с банковского счета пользователя.

5. Если все прошло успешно, сервис регистрирует окончание поездки и возвращает номер заказа (поездки) и конечную стоимость.

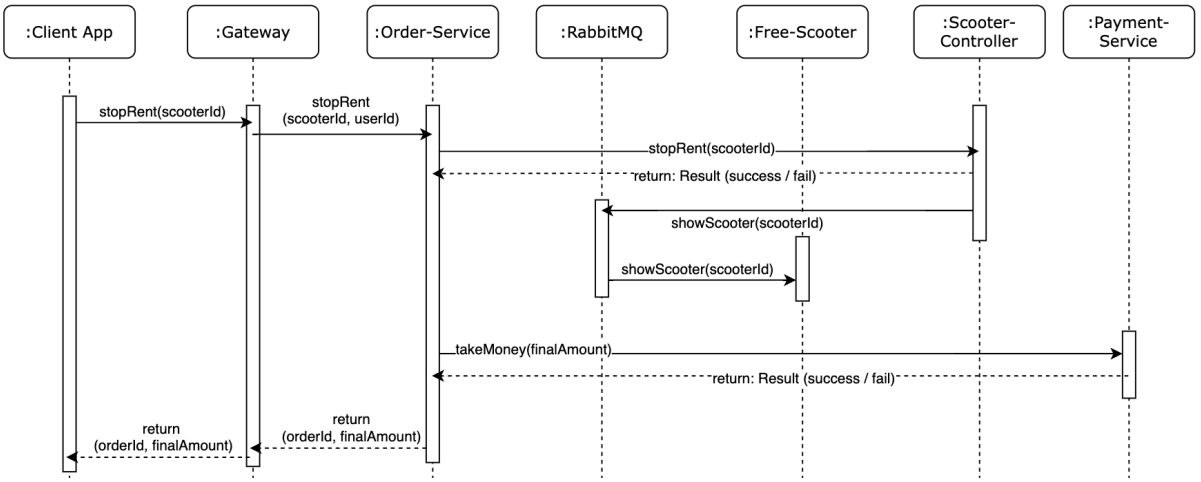

Рис. 6: Диаграмма последовательностей: окончание аренды

6. Особенности реализации

6.1. Асинхронное общение через RabbitMQ

RabbitMQ поддерживает несколько типов асинхронного общения сервисов, которые поддерживают разное количество получателей и отправителей. В данном проекте были использованы лишь два типа общения: Publish/Subscribe (один отправитель, много получателей) и Routing (много отправителей, много получателей).

Разница между Publish/Subscribe и Routing заключается в том, что первый тип общения рассылает сообщения всем подписанным на него сервисам, а второй тип общения отправляет сообщения только тем сервисам, которые подписаны на определенный тип сообщений.

В данном проекте Publish/Subscribe был использован в отсылке пароля от сервиса генерации пароля до сервиса аутентификации и авторизации, а Routing был использован, например, при общении между сервисом контроля самокатов и сервисом отображения доступных самокатов.

6.2. Контейнеризация

Контейнеризация сервисов является эффективным способом упрощения процесса развертывания и управления приложением, а также обеспечения его высокой доступности и масштабируемости. Для контейнеризации сервера были использованы Docker [10] и Docker-Compose [11].

Были созданы Docker-образы [12] для каждого сервиса, в которых были установлены все необходимые зависимости и настройки.

Далее был использован Docker-Compose для запуска всех сервисов вместе и автоматической настройки их взаимодействия. Docker-Compose позволил легко масштабировать приложение, добавляя сервисы по мере необходимости.

6.3. Аутентификация на основе областей видимости (scopes)

Не все пользователи должны пользоваться всеми конечными точками веб-сервера. Например, добавление новых самокатов, скрытие и показ самокатов недоступны обычным пользователям. Ими могут пользоваться только администраторы или менеджеры, то есть люди с определенными

дополнительными правами в системе. Для регуляции такого доступа была использована аутентификация на основе областей видимости (scopes).

Scopes – это механизм, который позволяет определить, какие ресурсы и данные могут быть запрошены приложением от пользователя. Scopes были использованы для ограничения доступа к данным пользователя, а также для управления правами доступа.

Например, для ограничения возможности пользования добавлением самокатов было добавлено новое разрешение "addscooter:api" в сервис аутентификации. Если пользователь имеет право на добавление нового самоката, то в его токен доступа приписывается возможность "addscooter:api".

Название области видимости (scope)	Описание области видимости
customer:api	Доступ к использованию серверного api для аренды самокатов, получения личной информации и т.д
addscooter:api	Доступ к добавлению новых самокатов в систему
chargescooter:api	Доступ к сокрытию и показу самокатов для пользователей

Таблица 2: Описание областей видимости.

Использование scopes позволило более точно управлять правами доступа пользователя.

Заключение

В ходе работы над проектом были достигнуты следующие результаты.

- Произведен анализ аналогичных сервисов: Woosh, Яндекс. Выявлены их преимущества и недостатки.
- Разработаны требования к системе.
- Разработана архитектура приложения. Выбран архитектурный стиль и построена полноценная микросервисная архитектура, включающая в себя следующие сервисы:
    - Gateway,
    - сервис генерации паролей,
    - сервис контроля самокатов,
    - сервис отображения свободных самокатов,
    - сервис контроля заказов,
    - сервис аутентификации и авторизации,
    - RabbitMQ.
- Реализована работа веб-сервера с применением выбранной архитектуры и современных подходов, а также были реализованы следующие сценарии использования:
    - добавление новых самокатов в систему,
    - постановка самокатов на зарядку,
    - начало и окончание аренды самоката.

Список литературы

[1] Самокаты Яндекс. URL: https://go.yandex/ru_ru/lp/rides/scooter

[2] Самокаты Whoosh. URL: https://whoosh-bike.ru/contacts

[3] Юрент. URL: https://urent.ru/

[4] Lime. URL: https://www.li.me/

[5] SK.RU. “Whoosh привлек от инвесторов $25 млн” URL: https://sk.ru/news/rossiyskiy-servis-arendy-elektrosamokatov-whoosh-privlek-ot-investorov-25-mln/

[6] Openiddict. URL: https://github.com/openiddict/openiddict-core

[7] PostgreSQL. URL: https://www.postgresql.org/

[8] EntityFramework Core. URL: https://learn.microsoft.com/en-us/ef/core/

[9] RabbitMQ. URL: https://www.rabbitmq.com/

[10] Docker. URL: https://www.docker.com/

[11] Docker-Compose. URL: https://docs.docker.com/compose/

[12] Docker-Image. URL: https://hub.docker.com/

[13] Ionescu V. M. The analysis of the performance of RabbitMQ and ActiveMQ //2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER). – IEEE, 2015. – С. 132-137.

[14] Hong X. J., Yang H. S., Kim Y. H. Performance analysis of RESTful API and RabbitMQ for microservice web application //2018 International Conference on Information and Communication Technology Convergence (ICTC). – IEEE, 2018. – С. 257-259.

[15] Eriksson H. E. et al. UML 2 toolkit. – John Wiley & Sons, 2003.

[16] Eriksson H. E., Penker M. Business modeling with UML //New York. – 2000. – Т. 12.

[17] Arlow J., Neustadt I. UML 2 and the unified process: practical object-oriented analysis and design. – Pearson Education, 2005.

[18] Scooter Github Repository. URL: https://github.com/yassinezaoudi/scooter