Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

КУРСОВОЙ ПРОЕКТ

по теме:

**Разработка клиент-серверного desktop-приложения для поддержки бизнес-процессов компании**

по курсу Базы данных

Студент:Победоносцев Г.А., 211-361

Преподаватель:Харченко Е.А.

Москва 2025

**Введение**

Предметной областью данного проекта является прокат велосипедов. Велосипед как вид транспорта всё больше набирает популярность в связи с растущим трендом на экологию. В городах строится велосипедная инфраструктура, такая как велодорожки, велосипедные светофоры, выделенные для велосипедного движения полосы, стоянки велосипедов. Однако в зимнее время года, в период гололедов, приобретенный в постоянное пользование велосипед просто занимает место в жилом помещении. Также покупка качественного велосипеда может нанести ущерб финансовому благополучию покупателя. Решением данных проблем является прокат велосипедов.

Проблемой организации бизнеса по прокату велосипедов является тот факт, что клиент, придя в магазин, может не обнаружить в наличии велосипед той модели, которая ему нужна, что будет являться упущенной прибылью для владельца бизнеса по прокату. Для решения данной проблемы принято решение разработать информационную систему для хранения данных о бронировании и аренде велосипедов.

**1 Постановка задачи**

Цель работы – разработать простое оконное клиент-серверное приложение, позволяющее пользователю посредством графического интерфейса и согласно предоставляемым ему правам доступа обрабатывать данные, хранящиеся на сервере и моделирующие некоторую предметную область – прокат велосипедов.

Предметная область выбрана случайным образом из предложенных вариантов при помощи генератора случайных чисел (см. Рисунок 1).

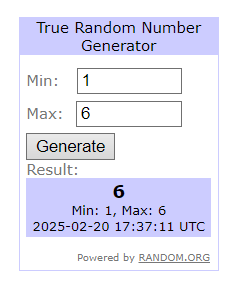


Рисунок 1 – Выбор варианта

Задачами данной работы являются:

1. Зафиксировать требования к системе и описать их при помощи диаграммы вариантов использования;
2. Спроектировать схему базы данных, создать инфологическую, реляционную и физическую модели, а также CRUD-матрицу;
3. Спроектировать структуру кода и проиллюстрировать ее при помощи диаграммы классов;
4. Разработать приложения на основе проектной документации;
5. Указать специфику реализации приложения;
6. Продемонстрировать возможности приложения на нескольких неодносложных примерах;
7. Выявить достоинства и недостатки приложения, рассмотреть перспективы развития проекта.

**2 Проектирование и разработка**

Проектирование информационной системы включает в себя описание требований, проектирование схемы базы данных и проектирование структуры кода.

Требования к разработке системы:

1. Средства разработки – язык программирования Java, система управления базами данных MySQL, графическая библиотека – JavaFX.
2. Код программы должен быть размещен в системе контроля версий Git;
3. Структура кода программы должна строго соответствовать шаблону проектирования MVC (Model-View-Controller);
4. Каждая таблица базы данных должна удовлетворять нормальной форме Бойса-Кодда;
5. Должны быть предприняты исчерпывающие меры для обеспечения целостности данных при эксплуатации базы данных;
6. Должна быть реализована система регистрации и авторизации пользователей. Пользователь должен иметь возможность изменять свои регистрационные данные;
7. Для ограничения доступа пользователей к записям базы данных должна использоваться ролевая модель;
8. Для пользователей с различными привилегиями должен быть реализован соответствующий функционал в концепции CRUD (Create-Read-Update-Delete), доступный через графический интерфейс пользователя;
9. Запросы к базе данных должны быть параметрическими (с целью защиты от SQL-инъекций);
10. Результат запроса должен «оборачиваться» в объект специально разработанного класса согласно шаблону Singleton;
11. Должен быть создан исполняемый JAR-файл программы.

Требования к функционалу приложения:

1. Каждый клиент имеет номер и серию паспорта, идентификационный номер, имя и адрес;
2. Каждая модель велосипеда имеет уникальное название, тип (городской, дорожный, горный) и количество передач;
3. Каждый велосипед имеет уникальный идентификационный номер и модель;
4. У компании есть несколько различных магазинов, где можно забрать и вернуть велосипеды. Каждый из этих магазинов идентифицируется уникальным именем и имеет адрес (оба обязательны);
5. При бронировании необходимо знать следующее: какой клиент сделал бронирование, когда он заберет велосипед (день), какую модель велосипеда он хочет и где он заберет велосипед (магазин). Ведется учет о находящихся в аренде велосипедах;
6. При возврате велосипеда сохраняется дата возврата.

На основании вышеперечисленных требований была построена диаграмма вариантов использования (рисунок 2).

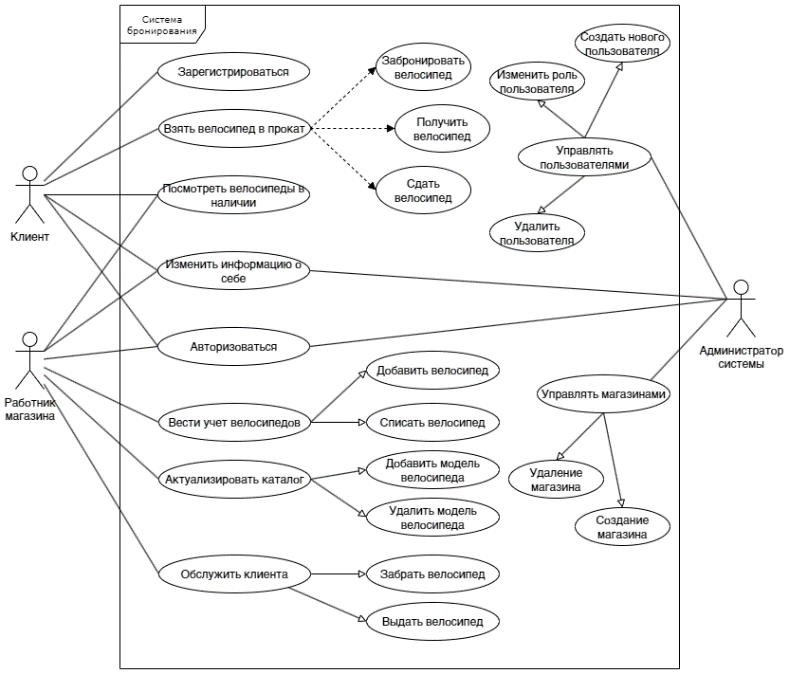


Рисунок 2 – Диаграмма вариантов использования

Было выделено 3 роли – клиент, работник магазина и администратор системы. На основании требований разделены допустимые сценарии работы. Клиент может осуществлять действия, направленные на работу в системе (регистрация, авторизация, изменение информации) и на прокат велосипеда. Причем все действия по прокату велосипеда являются обязательными элементами бизнес-процесса, так как нельзя забрать велосипед, не забронировав, или нельзя его получить и никогда не возвращать.

Роль работника магазина предназначена для выдачи клиентам велосипедов, а также актуализации информации по велосипедам. Зарегистрироваться самостоятельно работник магазина не может – для него аккаунт создает администратор.

Роль администратора заключается в управлении доступом к системе, а именно созданию, редактированию и удалению учетных записей пользователей системы, в том числе создания новых учетных записей с ролью администратора. Помимо этого, администратор имеет возможность редактировать системный справочник с перечнем магазинов.

Далее была создана инфологическая модель системы. На ней были описаны основные объекты, участвующие в бизнес-процессе по прокату велосипедов (Рисунок 3). Как можно заметить, помимо тех полей, которые были указаны в требованиях к системе, всем пользователям было добавлено поле «Пароль», по которому будет осуществляться вход в систему.

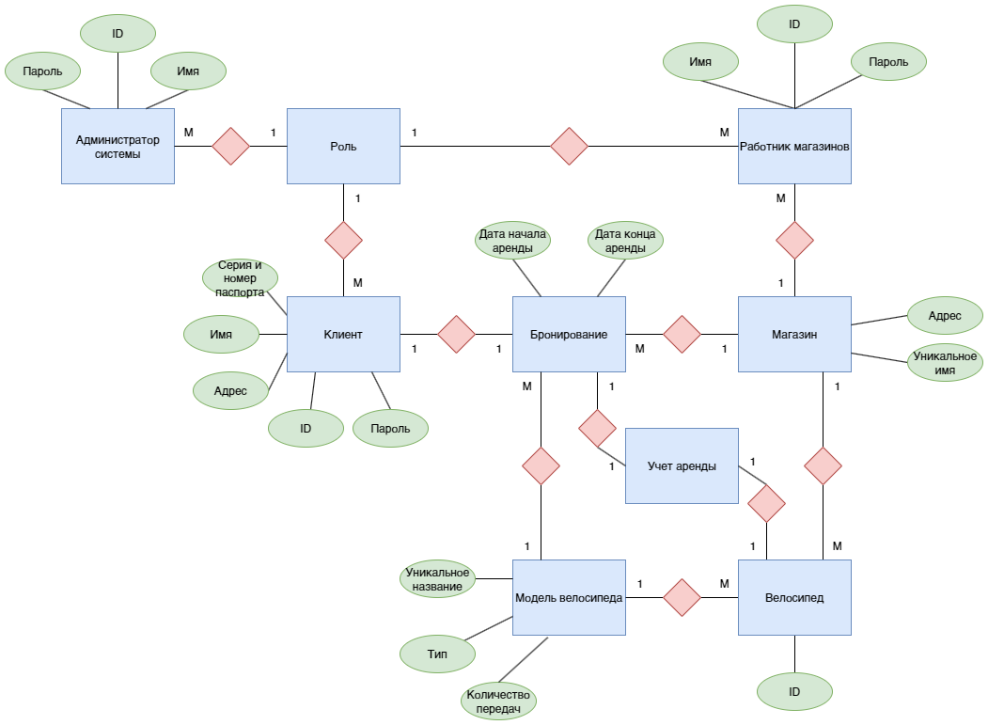


Рисунок 3 – Инфологическая модель базы данных

На основе инфологической модели была построена реляционная модель, в которой каждая таблица имеет нормальную форму Бойса-Кодда (Рисунок 4). Были определены первичные и вторичные ключи. Также был добавлен суррогатный ключ для объекта бронирования.

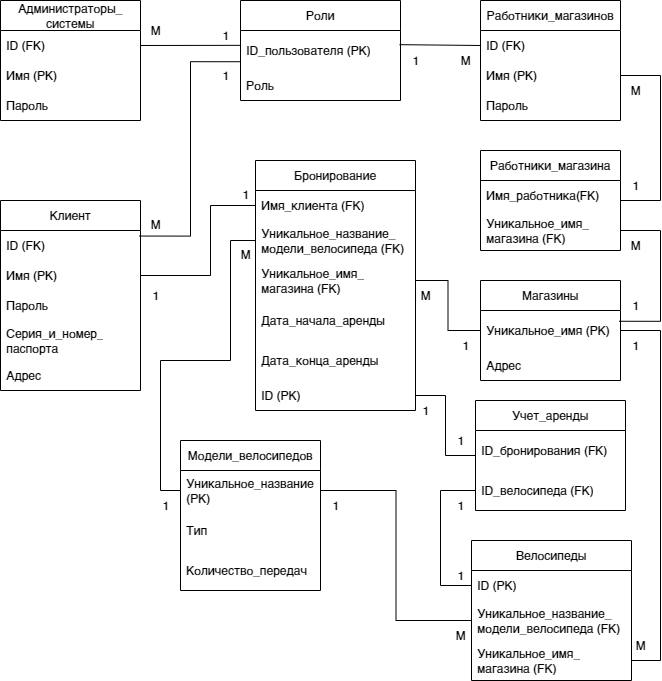


Рисунок 4 – Реляционная модель базы данных

Далее были определены типы каждого атрибута, на основе чего создана физическая модель базы данных (Рисунок 5).

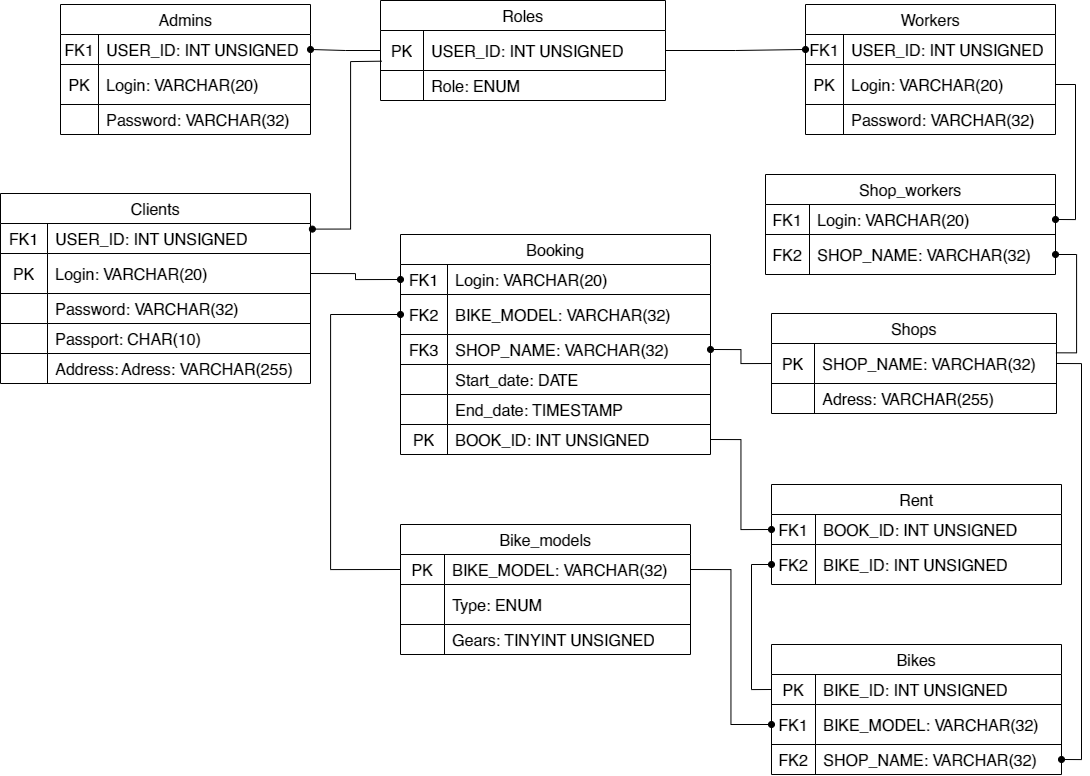


Рисунок 5 – Физическая модель базы данных

На основе пересечения таблиц БД и сценариев использования была составлена CRUD-матрица (Рисунок 6).

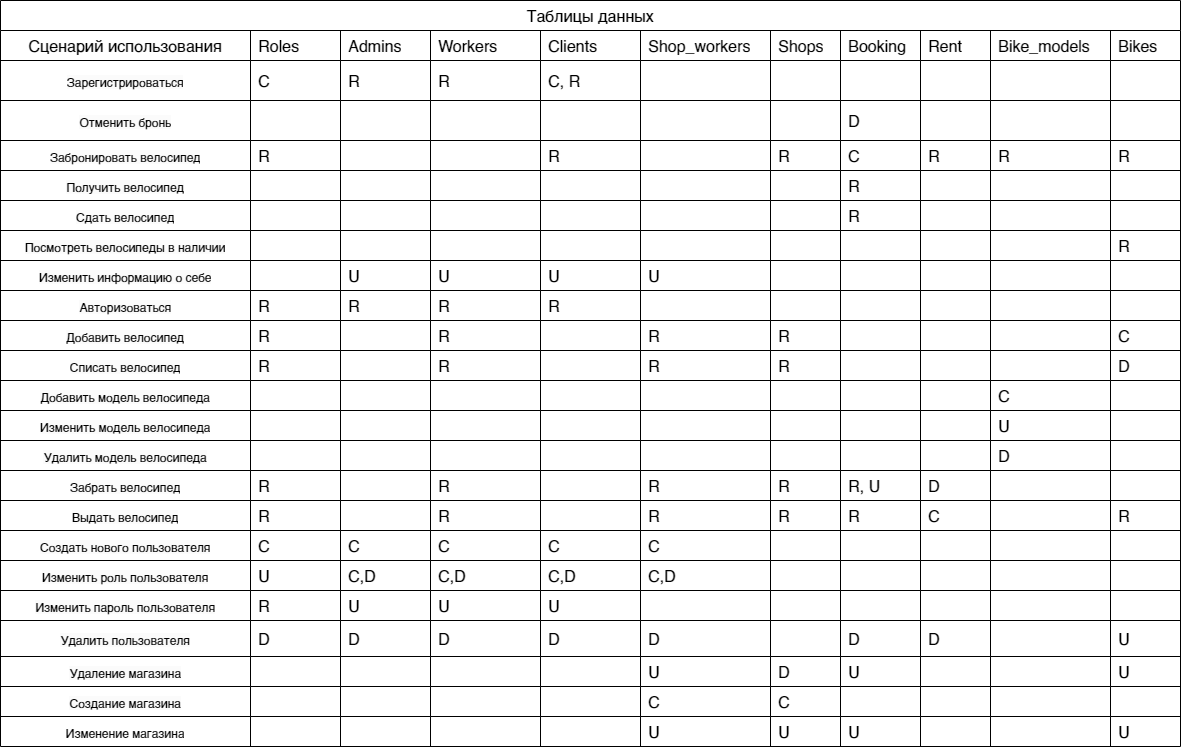


Рисунок 6 – CRUD матрица

Важно было обеспечить, чтобы для каждого FK при создании объекта было указано чтение таблицы, на основе которой заполняется какое-либо значение. Так же по итогу заполнения матрицы можно отметить, что абсолютно для каждой таблицы, кроме таблицы учета аренды, в столбце есть минимум по одной операции каждого типа. Для учета аренды нет сценария обновления, так как аренда имеет всего 2 состояния – велосипед либо в аренде, либо нет.

Создание CRUD матрицы позволило выявить ряд отсутствующих сценариев использования, которые не были предусмотрены на первом этапе проектирования системы. Были добавлены сценарии изменения ряда объектов, а также действие отмены брони, являющееся альтернативным получению велосипеда (Рисунок 7).

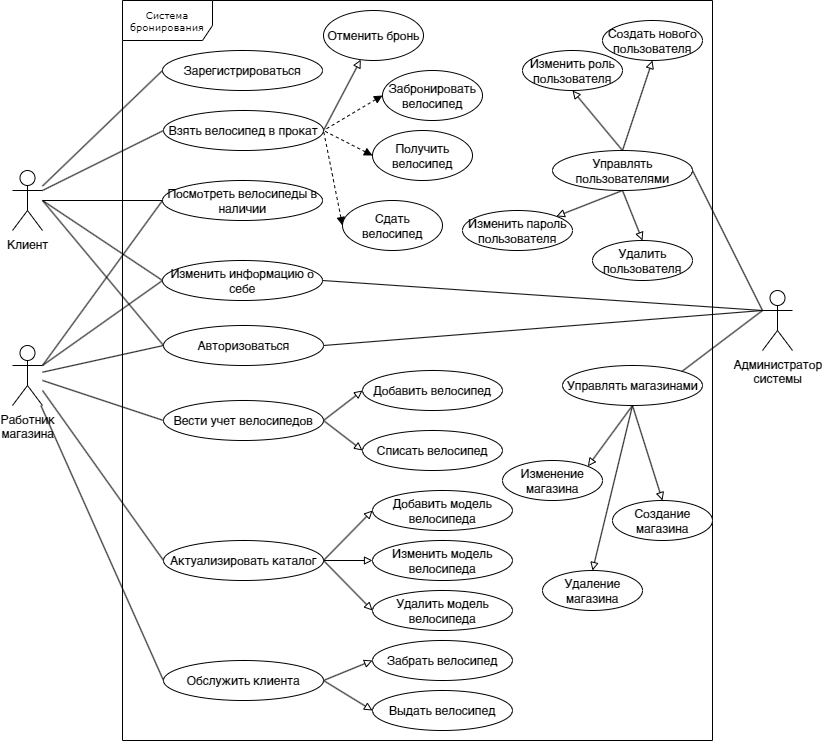


Рисунок 7 – Уточненная диаграмма вариантов использования

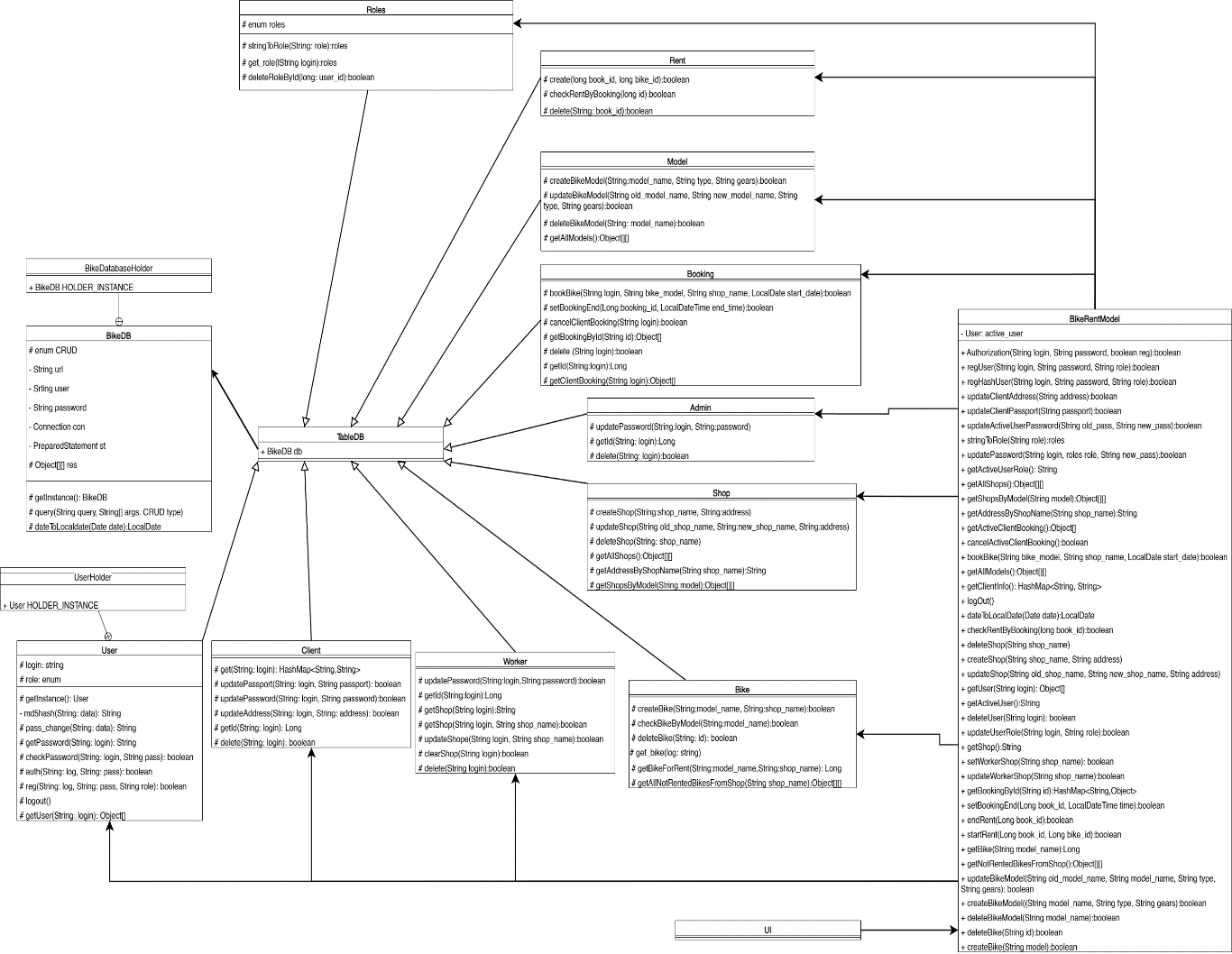
Далее параллельно с реализацией программы разрабатывалась диаграмма классов (Рисунок 8). 

Рисунок 7 – Диаграмма классов

По итогу была реализована MVC архитектура. В качестве модели выступает пакет, содержащий класс обращения к базе данных BikeBD, оформленный в паттерн Синглтон, а также ряд классов для составления запросов и их обработки. Помимо BikeBD синглтоном является класс User, представляющего активного пользователя системы.

Также был создан класс BikeRentModel представляющий из себя реализацию паттерна Фасад. Это единственный класс в модельной части, имеющий у себя публичные методы. Такая реализация позволяет сделать классы обработки запросов БД независимыми друг от друга, а также избежать внешнего обращения напрямую к тем или иным классам.

Пользовательский интерфейс же разбит на fxml-страницы, отвечающие за отображение информации, и на группы контроллеров, отвечающие за обработку ввода пользователя. Контроллеры вызывают различные методы класса BikeRentModel, чтобы получать информацию из БД. При этом класс BikeRentModel никак ответно не взаимодействует с пользовательским интерфейсом, то есть бизнес-логика полностью отделена от представлений, что соответствует главной идее архитектуры MVC.

Также стоит отметить, что для эффективного управления переходом между экранами был создан Синглтоновый класс ScreenController, обеспечивающий управление страницами. Это позволило избежать хранения ссылок на экраны внутри разных контроллеров.

**Список литературы и интернет-ресурсы**

1. <https://www.drawio.com>
2. <https://filldb.info/dbgenerator>
3. <https://habr.com/ru/articles/321050/>