Федеральное государственное образовательное бюджетное
учреждение высшего образования
**«Финансовый университет при Правительстве
Российской Федерации»**

Департамент анализа данных, принятия решений и финансовых технологий

Курсовая работа по дисциплине
«Современные технологии программирования»
на тему:

«Разработка приложения-игры «Что? Где? Когда?» с использованием библиотек Spring Boot и JavaFX»

Выполнил:
студент ПИ19-3
Данилин А. А.

Научный руководитель:
Доцент, кандидат физико-математических наук
Корчагин С.А.

Москва
2021

Оглавление

Введение

«Что? Где? Когда?»- любимое интеллектуальное телешоу россиян с семидесятых годов двадцатого века и по сей день. За многие годы телеигра пережила множество перемен, сменила нескольких ведущих и бесчисленное количество игроков. Однако ключевые аспекты игры всегда оставались неизменными. Одним из них является формат проведения: знатоки собираются в тесной комнате, в окружении других команд, и вместе решают задачи, поставленные перед ними телезрителями. В период пандемии сохранять такой формат будет неразумно в связи с повышенным шансом заражения вирусом. Учитывая возраст большинства знатоков, об игре пришлось бы забыть вплоть до полной победы над болезнью.

В качестве демонстрации одного из вариантов решения возникшей проблемы был разработан прототип на языке программирования Java, содержащий в себе 2 решения, взаимодействующих между собой с помощью архитектуры REST: серверная часть с использованием фреймворка Spring Boot, а также клиентская часть: с графическим интерфейсом с использованием библиотеки JavaFX. Оба решения будут использовать модель MVC, которая разграничивает управляющую логику программы на отдельные компоненты, а за счёт применения Java и виртуальной машины JVM решение будет кроссплатформенным.

Постановка задачи

В соответствии с выбранной темой требуется разработать клиент-серверное решение с использованием библиотек Spring Boot для сервера и JavaFX для GUI клиента в виде пользовательских классов и таблиц для СУБД.

Со стороны клиента необходимо разработать несколько окон и логику переходов пользователей между ними, а также их дизайн и расположение элементов интерфейса для взаимодействия с пользователем.

Со стороны бекенда необходимо использовать ORM для связи Spring с СУБД, а также модель MVC для отдельного расположение контроллеров, сервисов и репозиториев с логикой таблиц СУБД.

Решение не должно завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.

Описание предметной области

Предметной областью автоматизации является приложение игра «Что? Где? Когда?». Логика проста: администратор задает список вопросов и команд- участников. Далее капитаны команд получают от администратора уникальный код, который он вводит в окне выбора команды, чтобы получить право начать игру от имени своей команды. После запуска игры у капитанов будет 60 секунд для ответа на вопрос, и возможность апеллировать решение программы по их ответу, если они не согласны с результатом. Апелляции позже рассматривает администратор.

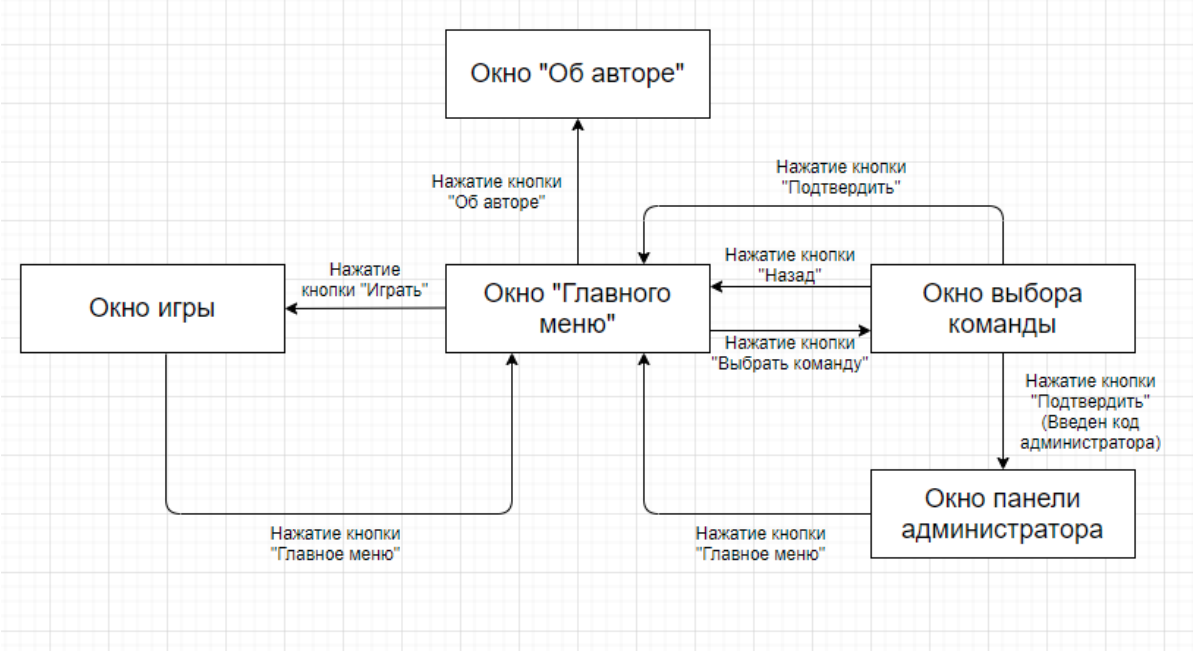
Актуальность автоматизации

Автоматизация проведения игры в онлайн формате позволяет сразу нескольким командам проводить игру и сразу узнавать результаты команд-противников. Также, в таком формате, участники остаются в безопасности за отсутствием необходимости собираться вместе для проведения игры.

Алгоритмические решения

Клиент

На рисунке ниже представлена схема перехода между формами. При обновлении данных вызывается запрос к серверу, во время которого осуществляется проверка на соединение с сервером, в случае отсутствия которого выдается сообщение об ошибке и начинается ожидание его появления. В случае восстановления связи с сервером, происходит автоматическая синхронизация всех данных и подгрузка обновленных форм.



(Переходы пользователя между формами в программе- клиенте)

При запуске программы осуществляется соединение с сервером и получение данных от сервера. В случае отсутствия соединения с сервером выдается ошибка и начинаются попытки повторного подключения и переход к главному окну приложения. В случае успеха происходит обновление форм, и пользователь получает возможность взаимодействия с формой. После, пользователь должен выбрать команду. При переходе к форме выбора

команды, открывается окно с полем для ввода, где пользователь вводит уникальный код своей команды. При получении кода, происходит проверка состояния команды. Если она допущена к игре и еще не сыграла, происходит переход к главному окну и открывается возможность начала игры. При начале, происходит обновление статуса команды и открывается окно игры, где у капитана показан вопрос и таймер на 60 секунд, в течение которого должен быть дан ответ. Если таймер истекает или дан неверный ответ, очко уходит зрителям, однако неверные ответы знатоки могут апеллировать. Верный ответ дает очко знатокам. В конце игры происходит обновление счета команды и список апелляций пополняется апелляциями, оставленными командой.

Сервер

При получении данных, сопоставляет их с существующей сущностью и обновляет соответствующее поле в базе данных Heroku Postgres.

При отправке, сервер получает данные из базы данных и отправляет клиенту в качестве jsonarray.

Описание интерфейса программы

В данном пункте речь пойдет об интерфейсе клиента, потому что это единственный модуль проекта, к которому есть доступ у пользователя. Для проектирования интерфейса была использована библиотека JavaFX.

Всего в клиенте представлено 5 окон:

-окно главного меню

-окно выбора команды

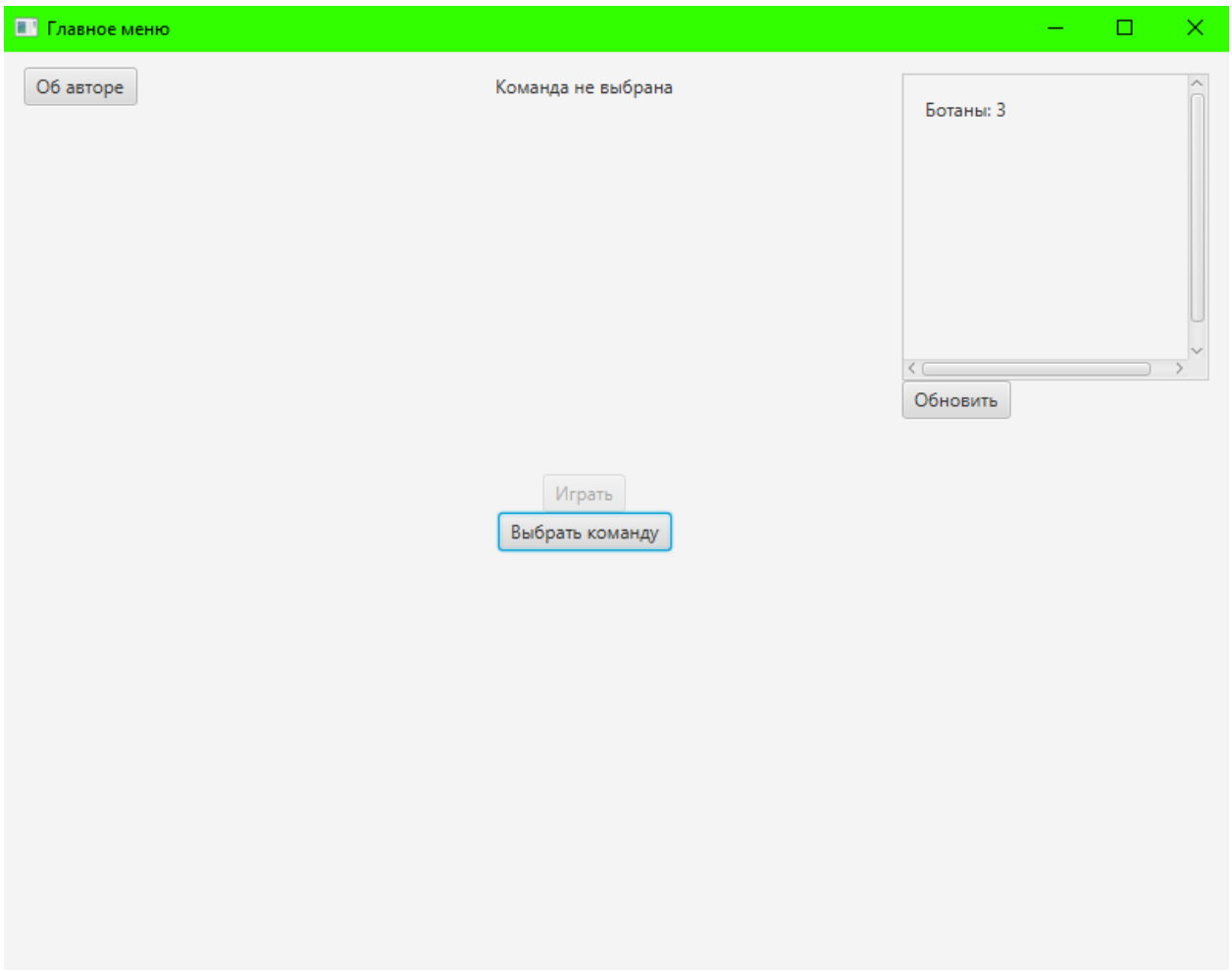
-окно игры

-окно панели администратора

-окно «Об авторе»

Окно главного меню

Позволяет осуществить переход в окно выбора команды и окно игры. Имеет поле, показывающее команду, выбранную пользователем, и таблицу лидеров. Если же пользователь еще не выбрал команду, то кнопка старта игры будет не активна, а в поле текущей команды будет соответствующее сообщение.

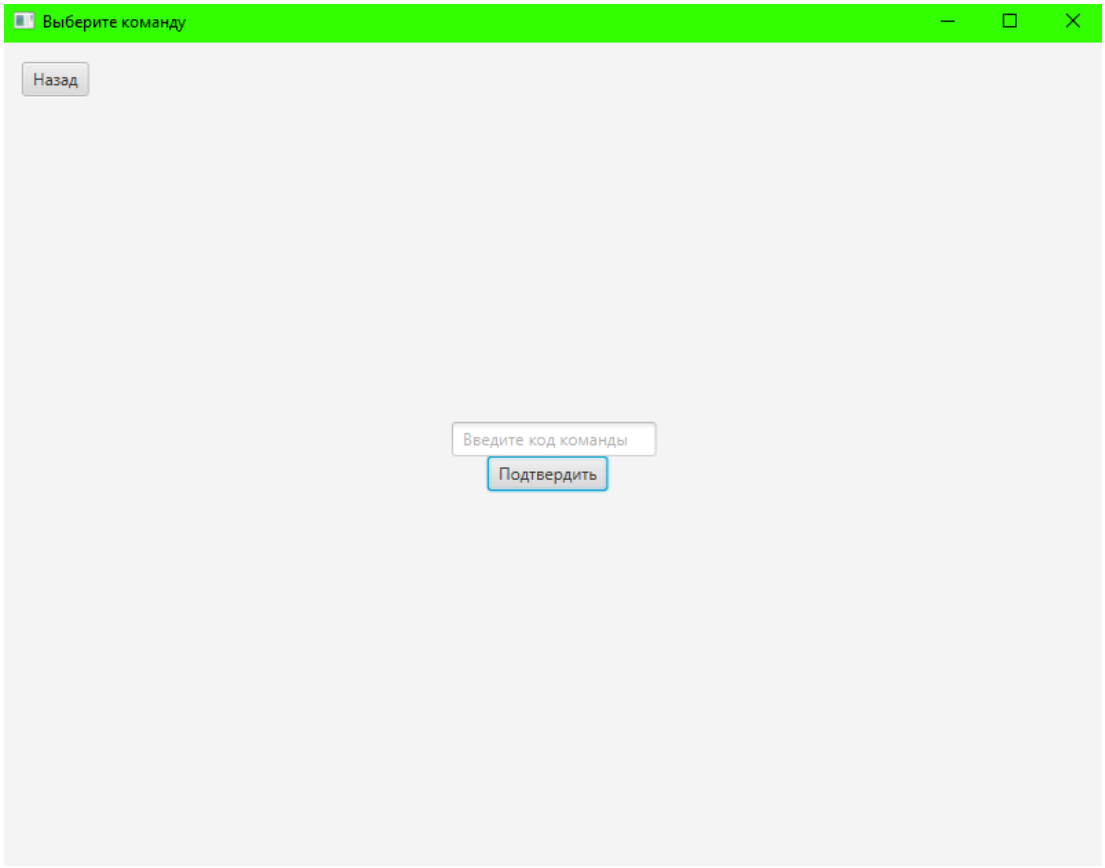


(Окно главного меню)

На форме существует проверка команды на предмет ее состояния.

Окно выбора команды

Запрашивает ввод уникального кода команды



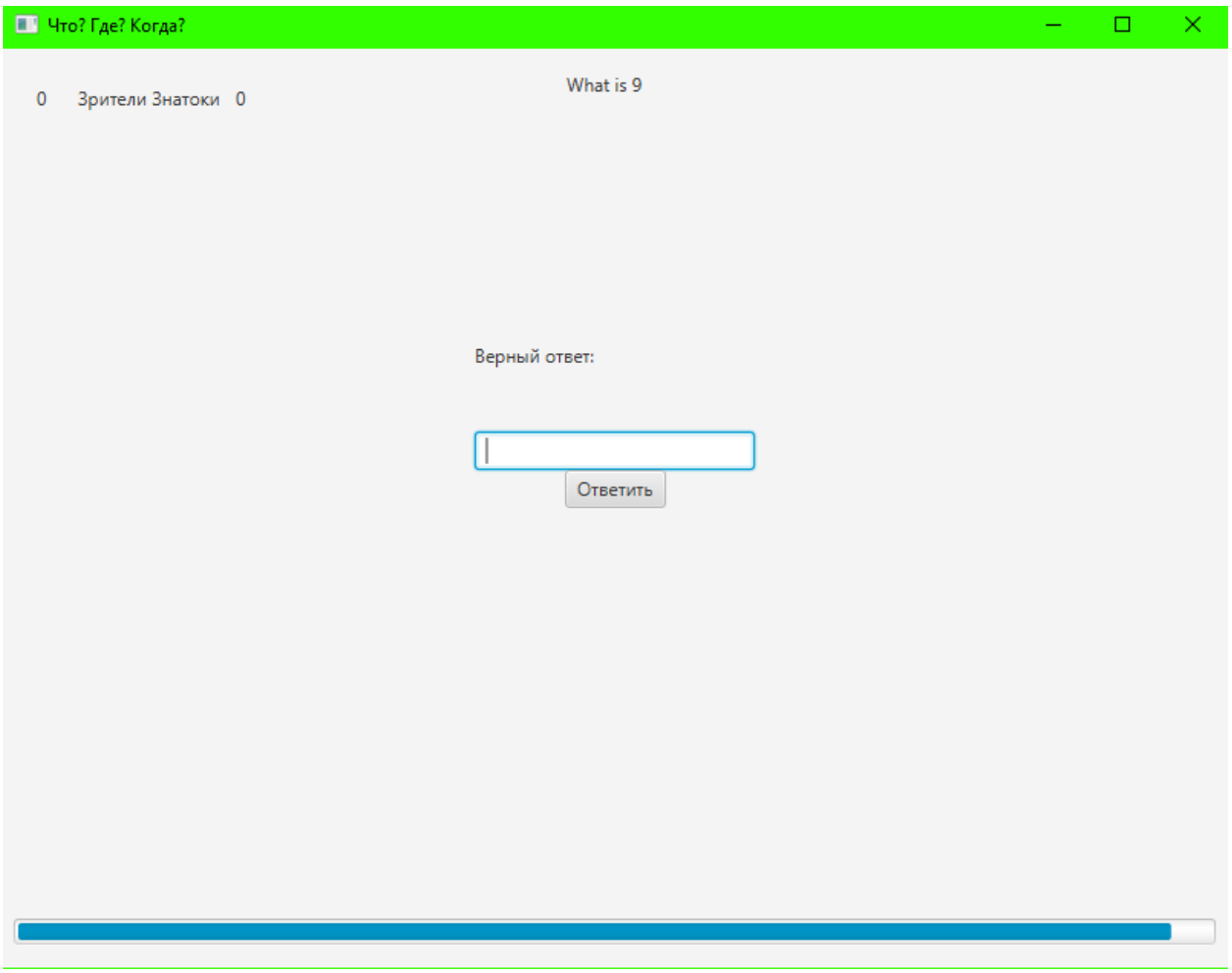
(Окно выбора команды)

При нажатии кнопки подтверждения происходит проверка существования данной команды и ее уровня доступа.

Если был введен код обычной команды и все условия были выполнены, происходит переход в главное меню и команда игрока устанавливается в соответствии с указанной в поле. Если был введен код панели администратора, происходит переход в окно панели администратора

Окно игры

В окне игры пользователю показан текущий счет, вопрос, поле для ответа и таймер.



(Окно игры)

При нажатии кнопки подтверждения ответа происходит проверка правильности введенного ответа. Пользователю показывается верный ответ и начинается отсчет 5 секунд. Если ответ пользователя верный, обновляется счет в пользу знатоков, иначе счет обновляется в пользу зрителей и у игрока появляется возможность запросить апелляцию. При нажатии кнопки запроса апелляции, в список апелляций добавляется новая запись, содержащая идентификатор команды, подавшей апелляцию, вопрос, ответ команды и верный ответ. Если ответ не был получен, очко уходит зрителям без возможности апелляции. Когда одна из сторон набирает 6 очков игра заканчивается, сервер получает обновленные данные по командам и апелляциям, а пользователь получает право выйти в главное меню. Если игра

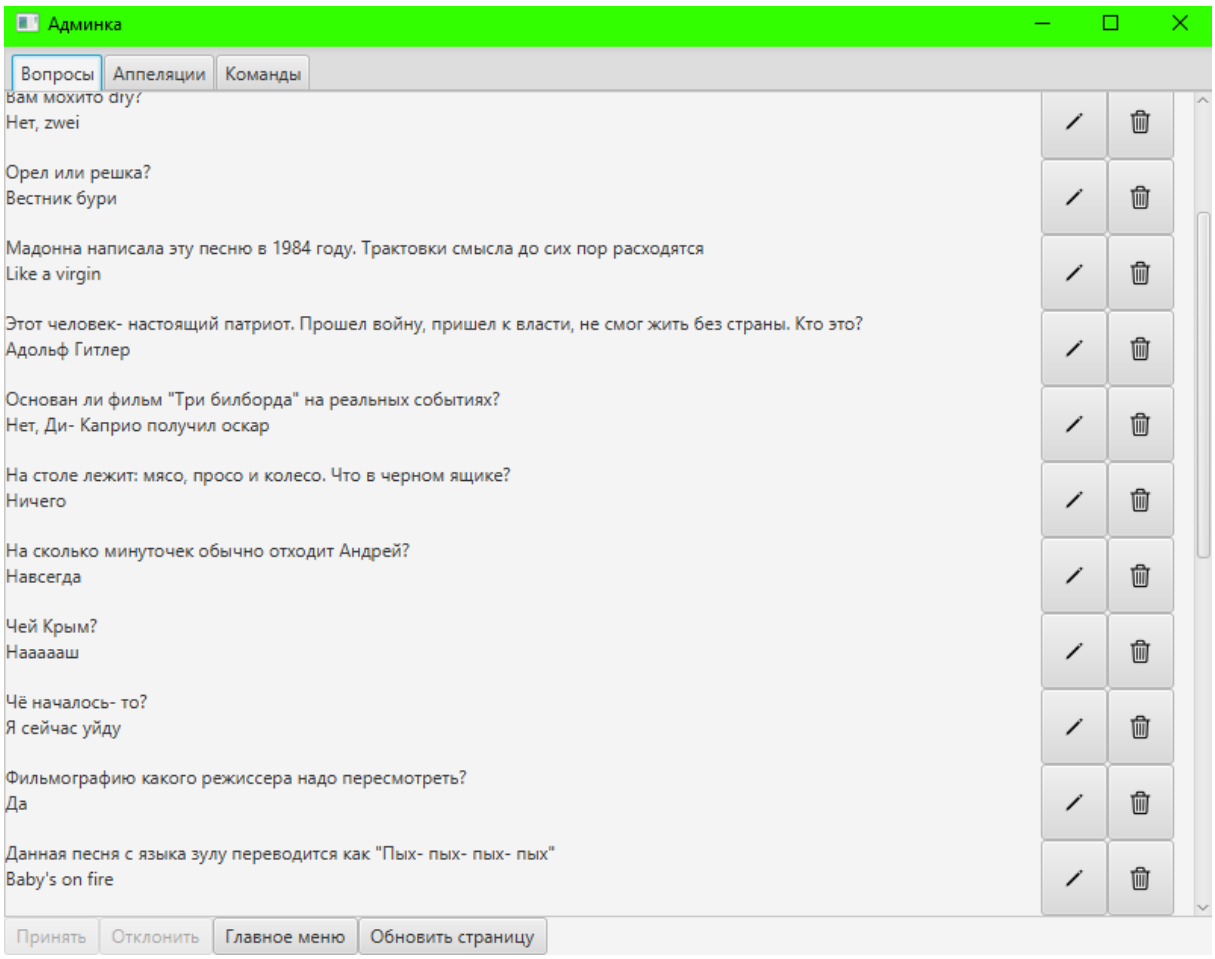
была завершена досрочно, счет команды становится 0, апелляции отклоняются.

Окно панели администратора

Отвечает за взаимодействие администратора и приложения. Позволяет предварительно настраивать вопросы, доступные для игры, количество команд, допущенных до соревнований, а также рассматривать апелляции, поданные капитанами во время игры.

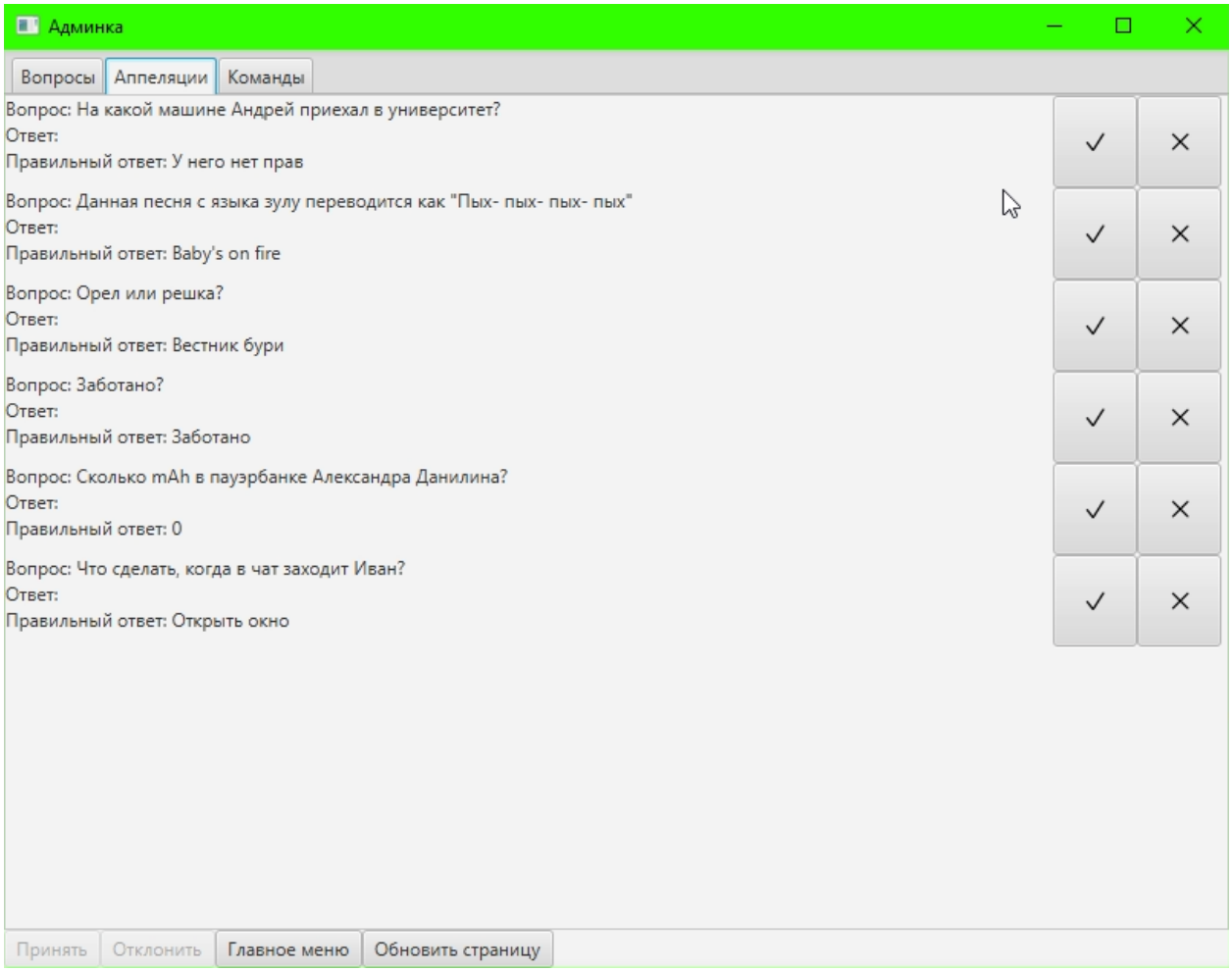
При нажатии кнопки «Принять» все изменения отправляются на сервер для последующей синхронизации с базой данных, при нажатии «Отклонить» все изменения отменяются и возвращаются к первоначальному виду. Кнопка «Главное меню» позволяет вернуться в главное меню, «Обновить страницу» повторно загружает данные с сервера. Кнопки «Принять» и «Отклонить» не активны до тех пор, пока пользователь не сделает изменений.

Все изменения выделяются зеленым цветом для наглядности.



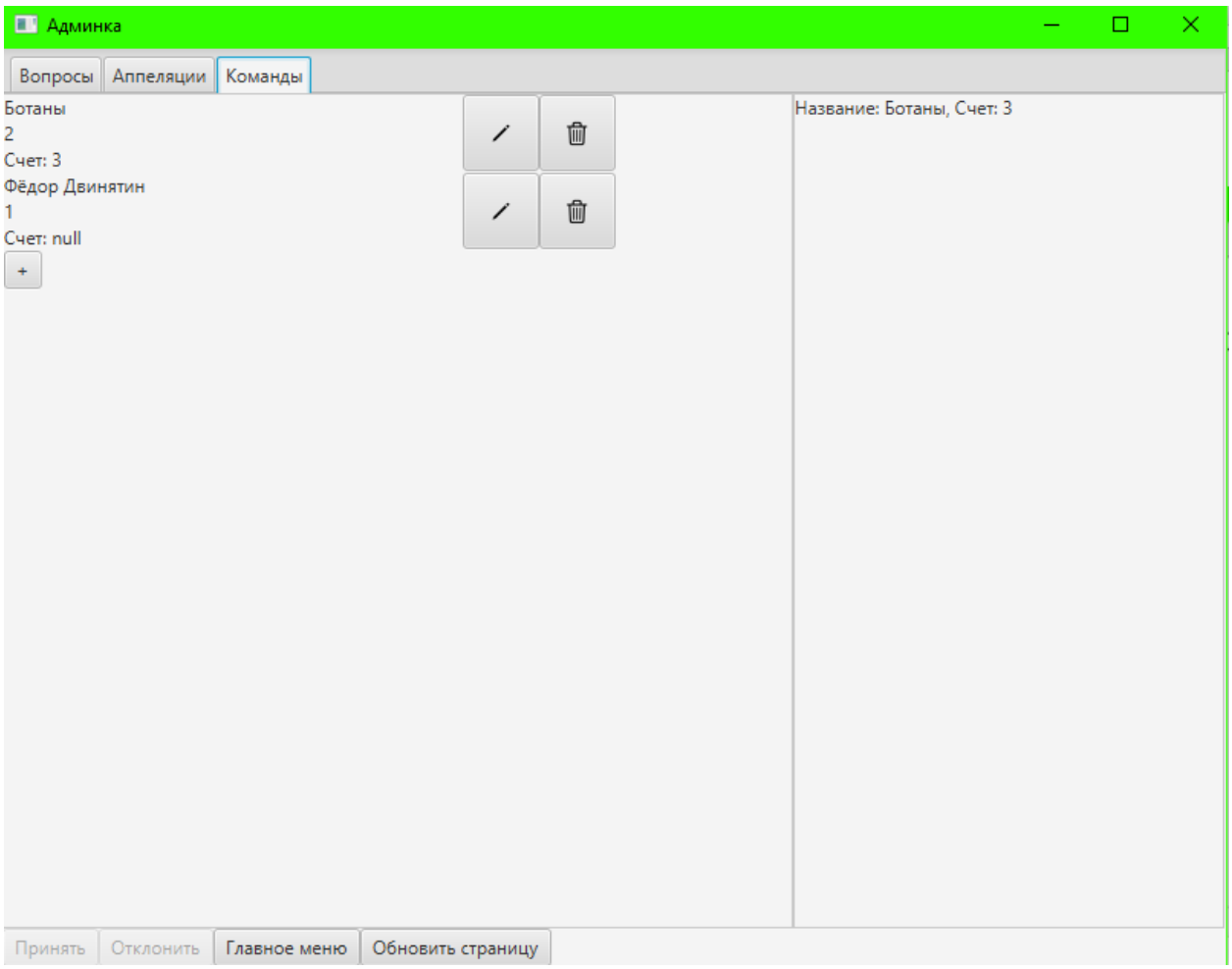
(Окно вопросов)

Позволяет редактировать список доступных вопросов



(Окно апелляций)

Позволяет рассматривать апелляции

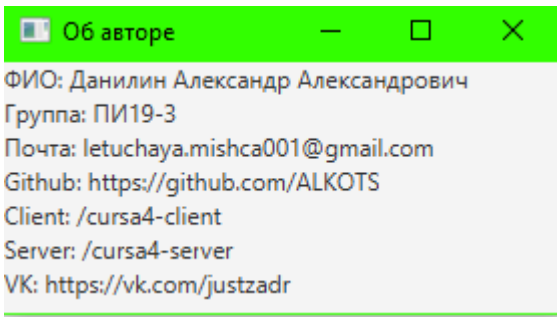


(Окно команд)

Позволяет редактировать список команд

Окно «Об авторе»

Окно «Об авторе» открывается параллельно с другими окнами и может существовать независимо. Содержит в себе информацию об авторе, в которую входят ФИО, группа, e-mail, id VK, а также ссылка на Github и репозитории проекта


Об авторе
ФИО: Данилин Александр Александрович
Группа: ПИ19-3
Почта: letuchaya.mishca001@gmail.com
Github: https://github.com/ALKOTS
Client: /cursa4-client
Server: /cursa4-server
VK: https://vk.com/justzadr

Состав приложения

Сервер

- Spring Web для создания web-приложений, в том числе RESTful, с использованием Spring MVC (Model-View-Controller). Использует сервер Apache Tomcat по умолчанию;
- Apache maven – основной сборщик проекта;
- Spring-boot-maven-plugin – плагин для сборки решения.

База данных

В качестве СУБД используется PostgreSQL субд Heroku Postgres и состоит из следующих таблиц и полей:

Questions- таблица вопросов. Содержит следующие атрибуты:

- id- уникальный идентификатор вопроса

- question- текст вопроса

- answer- текст правильного ответа

Appeals- таблица апелляций. Содержит следующие атрибуты:

-id- уникальный идентификатор вопроса

-team- уникальный идентификатор команды, подавшей апелляцию. Хранится в виде хэш кода.

- question- текст вопроса, на который была подана апелляция.

- answer- текст ответа, данного пользователем

- rAnswer- текст верного ответа

Teams- таблица команд

- id- уникальный идентификатор команды
- name- название команды
- accessKey- уникальный ключ доступа
- state- состояние команды (не допущена до соревнования, допущена или сыграла)
- score- счет команды
- email- адрес электронной почты команды

Клиент

В состав клиента входят следующие компоненты и сторонние библиотеки:

- JavaFX (javafx controls, javafx fxml, javafx collections, javafx scene, javafx stage)- библиотека отображения GUI и UI)
- Mashape (unirest)- библиотека обработки данных сервера
- json- обработчик json объектов для последующей конвертации в удобные контейнеры
- Apache maven- основной сборщик проекта, отвечает за подгрузку библиотек

Назначение и состав классов программы

Сервер

- Сущности (3 шт.)- таблицы СУБД
- Репозитории (3 шт.)- представление таблиц СУБД
- Cursa4ServerApplication- точка входа в программу

Клиент

- Классы- контроллеры (4 шт.)- содержат логику взаимодействия форм и пользователя, обработки действий
- Классы- утилиты (4 шт.)- содержат логику часто используемых функций и взаимодействий с сервером
- Точка входа в программу (1 шт.)- базовый класс запуска программы

Классы- контроллеры содержат в себе:

- main-menu-controller- контроллер взаимодействия пользователя и главного меню
- game-controller- контроллер матча, отвечает за взаимодействие пользователя и самой игры
- team-selector-controller- контроллер взаимодействия пользователя и меню выбора команды
- admin- controller- контроллер взаимодействия администратора и данных программы

Классы- утилиты

- my-timer- вспомогательный класс, создающий таймер, способный работать параллельно с потоком JavaFX и взаимодействовать со сценой
- SceneChanger- вспомогательный класс, загружающий новую сцену на основе запроса
- StageChanger- вспомогательный класс, открывающий новое окно со сценой, полученной от SceneChanger
- Unirests- вспомогательный класс, упрощающий взаимодествие клиента и сервера

Класс- Точка входа в программу

- Main- точка входа, содержит функционал получения данных от сервера и открытия главного меню

Заключение

В ходе проведения данной работы было создано приложение на базе Java, с использованием JavaFX для GUI и Spring Boot для сервера, позволяющее проводить любимую многими игру «Что? Где? Когда?» не выходя из дома.

Особенностями работы являются инструменты, использованные в ней, позволяющие используя PostgreSQL в качестве СУБД через Spring ORM и многопоточного взаимодействия JavaFX добиться оптимального взаимодействия пользователей внутри сообщества.

Созданное решение удовлетворяет всем требованиям и задачам: реализует CRUD-методы со стороны бекенда, а фронт осуществляет получение и отправку данных через RESTful API с помощью протокола http с последующим вводом/выводом данных на элементы управления графического интерфейса JavaFX.

Решение подготовлено к потенциальной модернизации. Так имеется возможность добавить поле e-mail в информацию о командах и автоматизировать отправку уникального кода сразу на почту пользователя.

Также, благодаря свойствам Java, приложению требуется минимум изменений и модификаций для выхода на рынок приложений для мобильных устройств.

Исходный код выполненной работы доступен по ссылке: https://github.com/ALKOTS/cursa4-client (клиент)

Список использованных источников

Учебная и научная литература

1. Володченкова Л.А., Козырев Д.В. Разработка серверной части программного приложения для удаленного хранения данных // МСиМ. 2020. №1 (53).
2. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. СОВРЕМЕННЫЕ ФРЕЙМВОРКИ ДЛЯ РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ // Интеллектуальные технологии на транспорте. 2020. №4 (24).
3. Гасанов Заурбек Зубаирович Анализ производительности многопоточных программ, написанных на языках Java и Go // Наука и образование сегодня. 2018. №6 (29).
4. Барабанов В.Ф., Донских А.К., Гребенникова Н.И., Кенин С.Л. ПОЛУЧЕНИЕ МЕТРИК JAVA-ПРИЛОЖЕНИЯ В КОНТЕЙНЕРАХ DOCKER // Вестник ВГТУ. 2020. №2.