**Серверная часть гипер-казуальной игры**

**Цель проекта**: разработка серверного ПО для гипер-казуальной игры.

**Актуальность** состоит в том, что наш сервер обеспечивает авторизацию пользователей, хранение их результата, многопользовательский режим, с помощью которого игроки смогут взаимодействовать друг с другом во время игры.

**Задачи** представлены на слайде.

Сервер написан на языке **Kotlin**. Это статически типизированный язык программирования от компании JetBrains. Он работает поверх виртуальной машины Java (JVM) и при компиляции компилируется в байт код. Поэтому мы можем запускать приложение на Kotlin везде, где установлена JVM. И, кроме того, можно компилировать код Kotlin в нативные бинарные файлы (с кодом, выполняемым непосредственно процессором устройства), которые будут работать без всякой виртуальной машины. Таким образом, круг платформ, для которых можно создавать приложения на Kotlin, чрезвычайно широк - Windows, Linux, Mac OS, iOS, Android

При создании сервера мы использовали **REST архитектуру.** В ней клиенты отправляют на сервер запросы для получения или модификации данных, а сервера отправляют клиентам ответы на их запросы.

Результаты игроков сохраняются в **реляционной базе данных** - это совокупность связанных между собой двумерных таблиц, в которых хранится информация об объектах. Каждая строка таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. Для работы с такими базами данных обычно используется язык SQL.

В основе сервера – паттерн **MVC.** Это шаблон программирования, который позволяет разделить логику приложения на три части и создавать их отдельно друг от друга:

* *Модель* получает данные от контроллера, выполняет необходимые операции и передаёт их в вид.
* *Представление* получает данные от модели и выводит их для пользователя.
* *Контроллер* – программный компонент, логика которого завязана на управлении прочими компонентами в зависимости от события. Он обрабатывает действия пользователя, проверяет полученные данные и передаёт их модели.

**Аналоги:**

* Leaderboard(он имеет проблемы с безопасностью)
* Photon(он работает только с языком Unity)