

УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Дисциплина «Облачные и туманные вычисления»

Этап 4 Лабораторной работы

Структурная схема с обоснованием

Студент

Ершова А. И.

P34302

Преподаватель

Перл О. В.

Санкт-Петербург, 2023 г.

Оглавление

Проблема	2
Общие сведения о приложении	3
UseCase диаграмма.....	4
Стек разработки.....	4
Объяснение замены некоторых сервисов	4
Архитектура приложения.....	6
Диаграмма развертывания.....	7
Диаграмма структуры Базы Данных	8
Настройка облачных сервисов.....	8
Создание кластера PostgreSQL	8
Создание Compute Cloud	10
Создание Smart Captcha	12

Проблема

Genshin Impact — очень популярная компьютерная игра в жанре action-adventure с открытым миром и элементами RPG, на данный момент насчитывает более 60 млн. игроков. В игре существует так называемая “гача-система”, в рамках которой игроки могут получить важные игровые артефакты только через казино-подобный механизм "молитв".

Однако перед игроками встает несколько сложных задач. Genshin Impact внедряет систему гарантов, при которой после определенного числа попыток игрок гарантированно получит 4- или 5-звездочный объект. Кроме того, игроки обнаружили наличие "мягкого гаранта", при котором вероятность выпадения 4- или 5-звездочных объектов часто возрастает немного раньше, чем предполагается официальными правилами.

В периодических обновлениях игры также увеличивается шанс получения определенных объектов. Этот сложный механизм вызывает у игроков желание оптимально распределять свои ресурсы, чтобы максимизировать шансы на получение желаемых артефактов.

В данном контексте, с учетом сложности расчетов, которые игроки должны проводить самостоятельно, появляется потребность в инструменте, который автоматизировал бы этот процесс. Мое намерение — создать приложение, которое на основе статистических данных игрока будет оценивать вероятность получения 4-звездочных и 5-звездочных объектов, упрощая игрокам принятие решений относительно использования ресурсов в игре.

Общие сведения о приложении

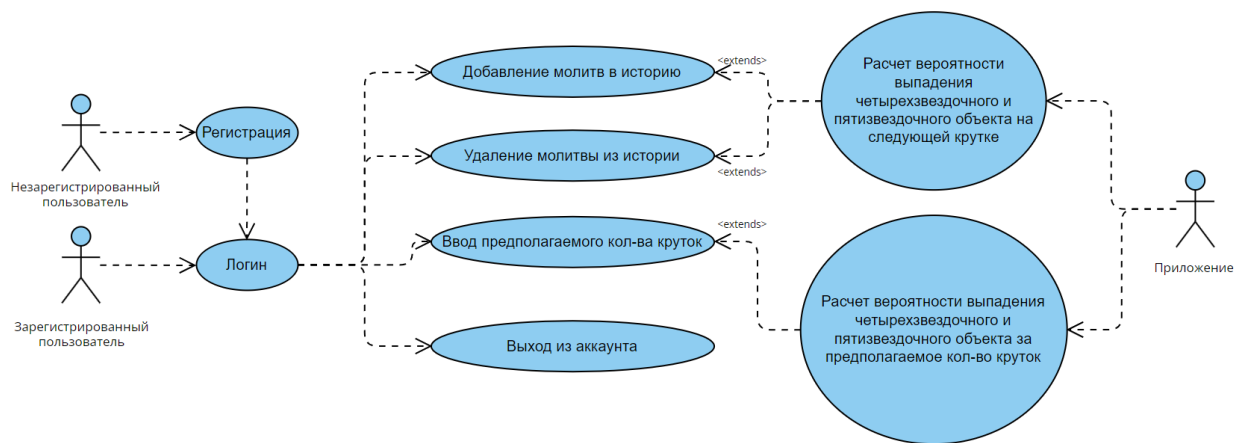
Приложение Prayers Predictor разработано с целью помочь пользователям игры Genshin Impact предсказывать возможный исход игровой механики «молитвы», чтобы более эффективно тратить ресурсы в игре.

Реализованы следующие функции:

1. Авторизация и Регистрация пользователей
2. Наличие капчи перед авторизацией
3. Добавление имеющихся у игрока молитв в Базу Данных с разделением на трехзвездочные, четырехзвездочные и пятизвездочные объекты
4. Удаление молитв из Базы Данных
5. Автоматический расчет вероятности выпадения четырехзвездочного объекта на следующей крутке
6. Автоматический расчет вероятности выпадения пятизвездочного объекта на следующей крутке
7. Расчет вероятности получения четырехзвездочного объекта с предварительным вводом количества предполагаемых круток
8. Расчет вероятности получения пятизвездочного объекта с предварительным вводом количества предполагаемых круток

Приложение и все его компоненты будут развернуты на облачном сервисе Yandex Cloud

UseCase диаграмма



Стек разработки

Сайт будет разработан на Java и будет использовать следующий стек технологий:

1. Backend App – Spring Framework (Boot, Security, Data) – фреймворк для разработки Spring-приложений
2. Frontend App – React – JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.
3. Yandex SmartCaptcha – сервис для верификации запросов, помогающий определить обращения пользователей и блокирующий ботов
4. Yandex Managed Service for PostgreSQL – сервис для управления кластерами объектно-реляционной СУБД PostgreSQL в инфраструктуре Yandex Cloud
5. Yandex Compute Cloud – сервис, предоставляющий масштабируемые вычислительные мощности для размещения, тестирования и прототипирования проектов.

Объяснение замены некоторых сервисов

В ходе реализации проекта решение насчет использования некоторых технологий были изменены

Замена Serverless Containers на Compute Cloud

Мое приложение изначально предполагало наличие фронтенд части, а в ходе более подробного чтения документации было выяснено, что с Serverless Containers можно общаться только посредством HTTP-запросов. К контейнеру нельзя никак подключиться, пробросить порт и т. д., чтобы посмотреть GUI. Соответственно в использовании Serverless Containers не было никакого смысла.

Замена YDB на PostgreSQL

Еще на 2-м этапе в качестве лучшей БД для моего проекта была выбрана PostgreSQL. Основные моменты:

1) В PostgreSQL есть возможность хранить геоданные, что важно для дальнейшего масштабирования моего приложения, т. к. у меня есть предположение, что распределение вероятностей выпадения тех или иных объектов может зависеть от региона проживания игрока.

2) YDB можно использовать только внутри облака Yandex Cloud. Повторюсь, что у меня есть желание масштабировать свой проект и выложить его в сеть, и в качестве хостинга с большой долей вероятности будет не Yandex Cloud. Соответственно использование YDB делает невозможным переезд с одной платформы на другую без переписывания кода.

Но на 2-м этапе после некоторого обсуждения с куратором было принято решение заменить PostgreSQL на YDB только из-за того, что у YDB есть возможность бесплатного использования, а я неправильно рассчитывала тарифы и думала, что за PostgreSQL придется платить из своего кармана.

При планировании проекта на 2-3 этапах для подключения YDB к Spring-приложению, я нашла 2 SDK:

1. <https://github.com/yandex-cloud/ydb-java-sdk/tree/master/spring-data-jdbc>

У данного sdk в README.md написано, что он устарел, собственно, он и не работал.

2. <https://github.com/ydb-platform/ydb-java-sdk>

При копировании предложенных зависимостей в pom.xml, а затем запуске примера, указанного по ссылке в README.md https://github.com/ydb-platform/ydb-java-examples/tree/master/basic_example импорты не работали и многих классов по указанным в примере путям не существовало при том, что зависимости точно были загружены правильно.

В ходе обсуждения данной проблемы с другими пользователями технологий YDB+Java выяснилось, что с этой же ситуацией столкнулись все. Из этого я сделала вывод, что данный SDK сломан, и необходимо переходить на другую БД.

После я узнала, что сумма за БД снимается не сразу за месяц, а посуточно, и поняла, что проблем с финансами изначально не было.

Архитектура приложения

Архитектура приложения реализовывает структуру MVC (Model-View-Controller). Данная структура позволяет разделить приложение на логические компоненты, упрощая управление и поддержку, упрощает внесение изменений и процесс масштабирования.

В соответствии с MVC приложение делится на 3 уровня:

1. View – Представление

На этом уровне находится клиентская часть приложения, которая разработана с использованием React.

React предоставляет компонентную модель для создания пользовательского интерфейса и обработки взаимодействия с пользователями

2. Controller – Контроллер

В приложении с помощью Spring Boot разработаны контроллеры, которые обрабатывают HTTP-запросы, поступающие от клиентской стороны. Контроллер получает запросы, взаимодействует с сервисами для выполнения бизнес-логики, а также обращается к базе данных при необходимости.

Контроллер также обеспечивает взаимодействие со Smart Captcha для проверки безопасности и аутентификации с помощью сервисов.

3. Model – Модель

Модель предоставляет бизнес-логику и данные приложения.

Данные хранятся в PostgreSQL с использованием Spring Data для доступа. Бизнес-логика приложения обрабатывает запросы и взаимодействует с БД при необходимости.

Для обмена данными между контроллером и моделью используются DTO-классы.

Диаграмма развертывания

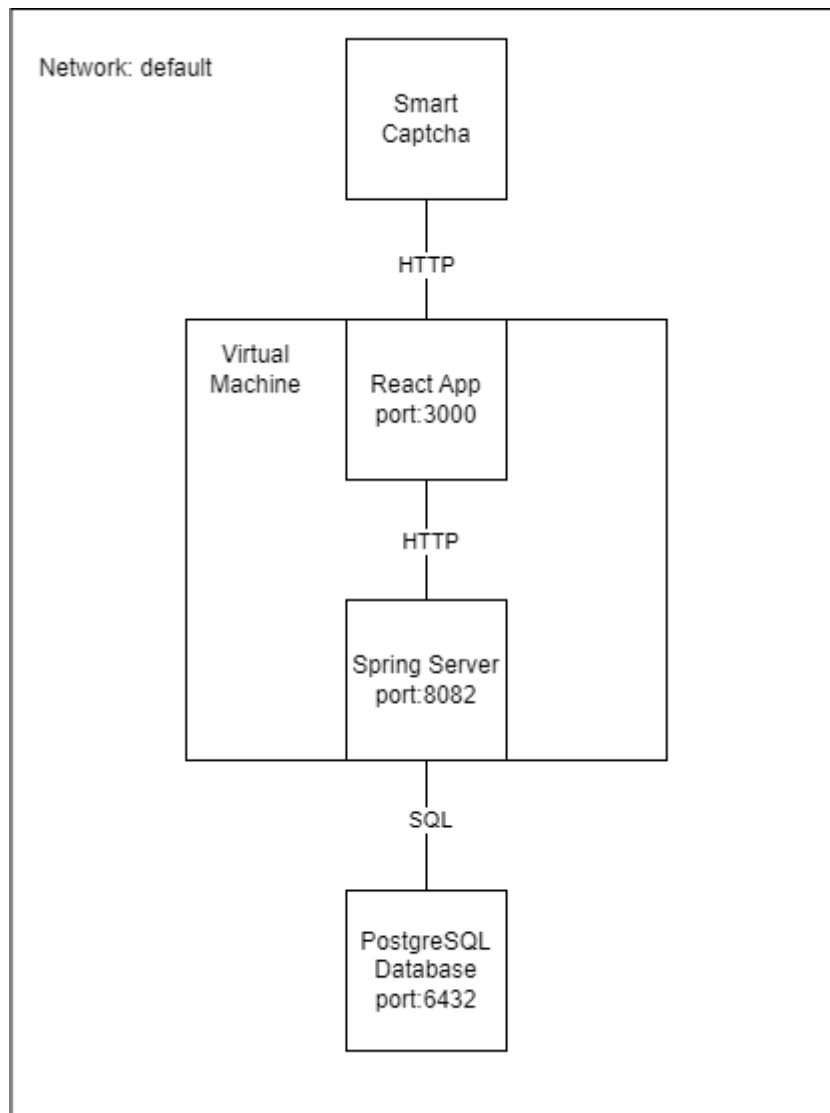
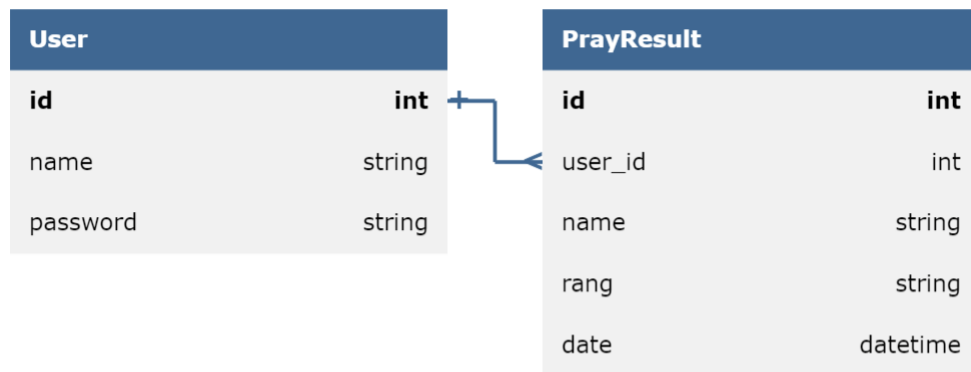


Диаграмма структуры Базы Данных



Настройка облачных сервисов

Создание кластера PostgreSQL

Создание кластера PostgreSQL

Базовые параметры

Имя кластера ?	<input type="text" value="postgresql243"/>
Описание ?	<input type="text"/>
Окружение ?	<div>PRODUCTION</div>
Версия ?	<div>15</div>
Метки	<div>Добавить метку</div>

Класс хоста

Платформа ?

Intel Ice Lake

Тип ?

cpu-optimized

standard

memory-optimized

c3-c2-m4

2 cores
vCPU

4 ГБ
Память

c3-c4-m8

4 cores
vCPU

8 ГБ
Память

c3-c8-m16

8 cores
vCPU

16 ГБ
Память

c3-c32-m64

32 cores
vCPU

64 ГБ
Память

c3-c40-m80

40 cores
vCPU

80 ГБ
Память

c3-c48-m96

48 cores
vCPU

96 ГБ
Память

Размер хранилища

network-ssd

network-hdd

local-ssd

network-ssd-nonreplicated

10 ГБ

10 ГБ

2053 ГБ

4096 ГБ

Макс. IOPS ?

Чтение 1000

Запись 1000

Макс. bandwidth ?

Чтение 15 МБ/с

Запись 15 МБ/с

База данных

Имя БД ?

gensh

Имя пользователя ?

ersio

Пароль ?

Сетевые настройки

Сеть ?

default

Группы безопасности ?

default-sg-enp6d30k69vunpkj5otm (default)

3 917,30 ₽ в месяц ?
Тарифы и цены
PostgreSQL. Intel Ice Lake. 100% vCPU 2 347,20 ₽
Публичный IP-адрес - PostgreSQL 172,80 ₽
Быстрое сетевое хранилище — PostgreSQL 130,10 ₽
PostgreSQL. Intel Ice Lake. RAM 1 267,20 ₽

Для подключения в Spring приложении изменяем application.properties

```
spring.datasource.url=jdbc:postgresql://c-c9q8Lo591870gpgk63m.rw.mdb.yandexcloud.net:6432/gensh
spring.datasource.username=ersio
spring.datasource.password=***
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.driver-class-name=org.postgresql.Driver
```

Создание Compute Cloud


Базовые параметры


Имя ?	<input type="text" value="cloud-vm"/>
Описание ?	<input type="text"/>
Зона доступности ?	<input type="text" value="ru-central1-b"/> v
Метки	<input type="button" value="Добавить метку"/>


Выбор образа/загрузочного диска


Операционные системы Container Solution Cloud Marketplace Пользовательские


Фильтр по операционной системе


 **Ubuntu** 22.04 ▼ ⓘ

 **Ubuntu OS Login** 22.04 LTS ▼ ⓘ

 **CentOS** 7 ▼ ⓘ

 **Debian** 11 ▼ ⓘ

 **Fedora** 35 ▼ ⓘ

 **openSUSE Leap** 15.4 ▼ ⓘ

Показать все продукты

Диски и файловые хранилища

Диски 1 Файловые хранилища

Ubuntu 22.04 LTS **Загрузочный** ...

Тип ⓘ HDD SSD SSD IO Нереплицируемый SSD

Размер ⓘ 8 ГБ 8 ГБ 8192 ГБ

Макс. IOPS ⓘ Чтение 300 Запись 300

Макс. bandwidth ⓘ Чтение 30 МБ/с Запись 30 МБ/с

Обязательно ставим галочку «Прерываемая», чтобы значительно снизить цену Виртуальной машины

Вычислительные ресурсы

Платформа ⓘ Intel Ice Lake ▼

vCPU 2 2 96

Гарантированная доля vCPU ⓘ 20% 50% 100%
Для тестирования прототипов без нагрузки или с минимальной нагрузкой.

RAM 2 ГБ 1 ГБ 8 ГБ

Дополнительно ☒ Прерываемая ⓘ

498,56 **₽** в месяц ▼ ?

[Тарифы и цены](#)

Intel Ice Lake. 20% vCPU — прерываемые VM

201,60 **₽**

Публичный IP-адрес

172,80 **₽**

Intel Ice Lake. RAM — прерываемые VM

100,80 **₽**

Стандартное сетевое хранилище (HDD)

23,36 **₽**

Создание Smart Captcha

Создание капчи

Настройки

Имя	<input type="text" value="captcha"/>
Основное задание	<div>Чекбокс Слайдер PREVIEW</div>
Дополнительное задание	<div>T Распознавание текста ▼</div>
Сложность ?	<div>Средняя ▼</div>
Отключить проверку домена	<input type="checkbox"/>
Список сайтов* ?	<div><input type="text" value="84.201.177.78"/> ×</div> <div>Добавить сайт</div>

Внешний вид

Стандартный	Серый	Темная тема	Синий

Настройка внешнего вида >

Создать

Отменить

Для подключения выполняем в терминале:

```
npm i -PE @yandex/smart-captcha
```

Прописываем импорт в React-приложении

```
import { SmartCaptcha } from '@yandex/smart-captcha';
```

Запуск приложения

```
ersio@cloud-vm:~/spring-app/target$ java -jar genshinProg-0.0.1-SNAPSHOT.jar
```

```
:: Spring Boot ::                                     (v3.0.3)
```

```
2023-12-10T18:49:01.981Z      INFO 5337 --- [main] c.g.genshinProg.GenshinProgApplication : Starting GenshinPro
gApplication v0.0.1-SNAPSHOT using Java 17.0.9 with PID 5337 (/home/ersio/spring-app/target/genshinProg-0.0.1-SNAPSHOT.j
ar started by ersio in /home/ersio/spring-app/target)
2023-12-10T18:49:01.992Z      INFO 5337 --- [main] c.g.genshinProg.GenshinProgApplication : No active profile s
et, falling back to 1 default profile: "default"
2023-12-10T18:49:03.411Z      INFO 5337 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Sprin
g Data JPA repositories in DEFAULT mode.
2023-12-10T18:49:03.479Z      INFO 5337 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Dat
a repository scanning in 58 ms. Found 2 JPA repository interfaces.
2023-12-10T18:49:04.131Z      INFO 5337 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized
with port(s): 8082 (http)
2023-12-10T18:49:04.142Z      INFO 5337 --- [main] o.apache.catalina.core.StandardService : Starting service [T
omcat]
2023-12-10T18:49:04.142Z      INFO 5337 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet en
gine: [Apache Tomcat/10.1.5]
```

Compiled successfully!

You can now view genshin in the browser.

Local: <http://localhost:3000>

On Your Network: <http://10.129.0.7:3000>

Note that the development build is not optimized. To create a production build, use `npm run build`.

```
webpack compiled successfully
```

