УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Дисциплина «Облачные и туманные вычисления»

Этап 3 Лабораторной работы

Структурная схема с обоснованием

Студент

Ершова А. И.

P34302

Преподаватель

Перл О. В.

Санкт-Петербург, 2023 г.

Оглавление

Общие сведения о приложении

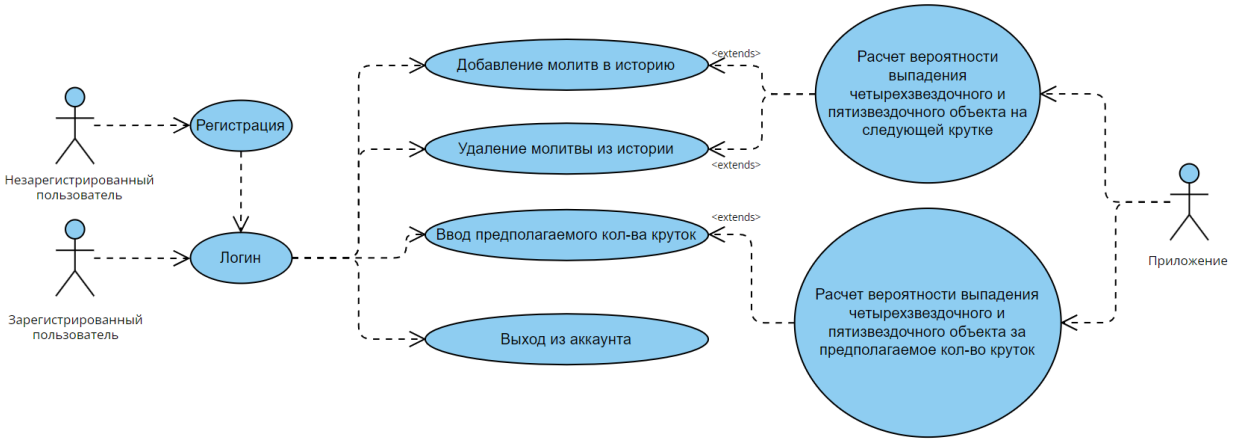
Приложение Prayers Predictor разработан с целью помочь пользователям игры Genshin Impact предсказывать возможный исход игровой механики «молитвы», чтобы более эффективно тратить ресурсы в игре.

Я планирую реализовать следующий функции:

1. Авторизация и добавление пользователя в Базу Данных
2. Внедрение функционала капчи в авторизацию
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:

Spring Framework (Boot, Security, Data) – фреймворк для разработки Spring-приложений

React – JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

2. YDB – распределенная отказоустойчивая Distributed SQL база данных с открытым исходным кодом.

3. SmartCaptcha – сервис для верификации запросов при регистрации

Архитектура приложения

Архитектура будущего приложения будет реализовывать 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 будет 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 будут разработаны контроллеры, которые будут обрабатывать 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.

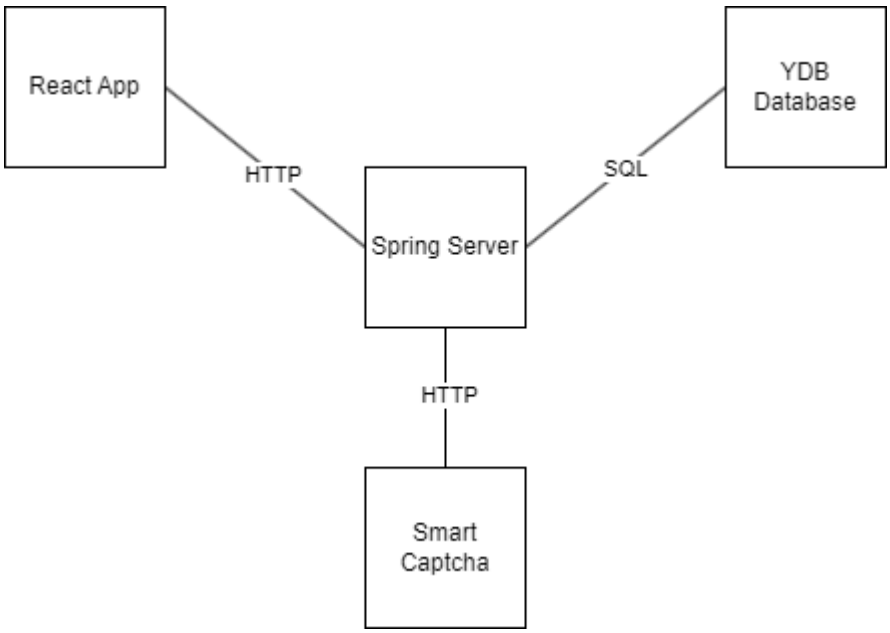
3. Model – Модель

Модель предоставляет бизнес-логику и данные приложения.

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

Для обмена данными между контроллером и моделью будут использоваться DTO-классы.

Диаграмма развертывания



Контейнерная диаграмма

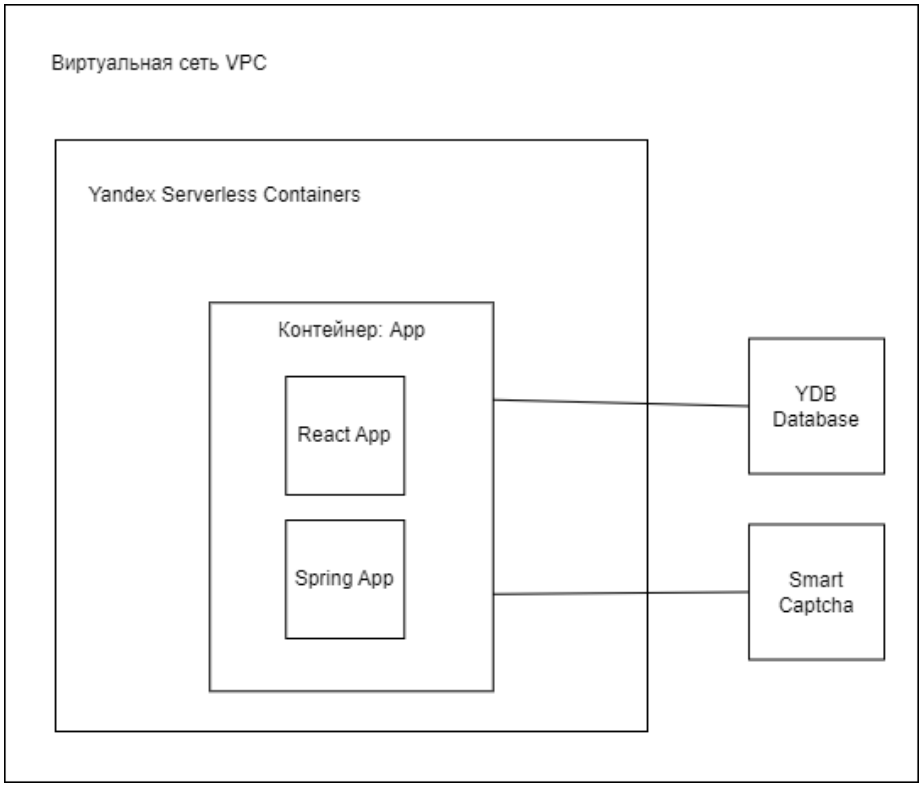
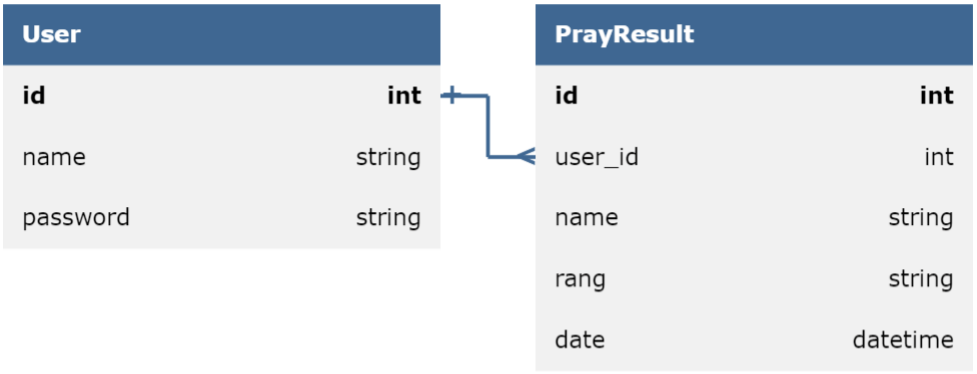


Диаграмма структуры Базы Данных



Целевая нагрузка системы

Ожидаемое количество пользователей:

Я ожидаю привлечь игроков Genshin Impact, которых по недавним подсчетам более 60 млн. Но я уверена, что не всех из них будет интересовать данное приложение, т. к. не все игроки настолько погружаются в игру. Также я не думаю, что первое время мое приложение будет популярно, поэтому остановлюсь на числе в 1000 пользователей.

Тип нагрузки:

Приложение будет обрабатывать операции, связанные с авторизацией, добавлением/удалением записей из БД, а также расчетами вероятностей выпадения объектов.

Операции пользователей:

Пользователи будут регистрироваться, авторизовываться и вводить свои молитвы в систему. Это предполагает операции вставки и чтения из БД.

Расчет вероятностей выпадения объектов будет выполнен в ответ на запросы пользователей

Прогноз по количеству операций:

Примерные оценки операций на пользователя в день:

- Регистрация/авторизация: 1 операция
- Ввод молитвы: 100 операций
- Расчет вероятности с определенным количеством круток: 10 операций

Предполагаемое количество пользователей: 1000 активных пользователей в месяц.

Спецификация по молитвам:

Я предполагаю, что средний пользователей будет вводить по 100 молитв в свою историю. Это значит, что в базе данных на 1 пользователя приходится по 100 записей в БД в день.

Если будет 1000 пользователей: 1000*100 = 100 000 записей в БД

Возможное масштабирование

В случае значительного роста числа пользователей, например, с 1000 до 100 000 система должна быть спроектирована для обработки более высокой нагрузки. Для этого можно произвести следующие изменения:

- Горизонтальное масштабирование серверов – добавление дополнительных серверов
- Кэширование – добавление кэширования для снижения нагрузки на базу данных. Это уменьшит количество запросов к БД и ускорит ответы на запросы пользователей.

Если объем данных начнет расти, например, если пользователь начнет добавлять вместо 100 по 1000 молитв в день, то система должна быть способной хранить и обрабатывать больше данных. Для этого можно произвести следующие изменения:

- Вертикальное масштабирование БД – увеличение ресурсов базы данных, включая производительность серверов БД и ее хранилища.
- Оптимизация запросов – произвести оптимизацию запросов к БД, чтобы они были более эффективными и работали быстрее

- Архивирование данных – можно реализовать механизм архивирования старых данных, которые редко используются, для более эффективного управления растущим объемом данных.

Со временем к действующему приложению будут обязательно добавляться новые функции, такие как оформление профиля, добавления аватаров, увеличение удобства взаимодействие с сайтом. Все это потребует увеличения количества таблиц в БД, увеличение объема хранимых данных. Для этого в дальнейшем можно будет произвести следующие изменения:

- Разработать систему мониторинга и аналитики, чтобы следить за работоспособностью системы и быстро реагировать на проблемы
- Масштабировать хранилище данных для хранения большего количества данных
- Разработать механизм для эффективного добавления и обработки новых данных, чтобы обеспечить внедрение новых функций: добавить версионирование данных, механизмы резервного копирования и восстановления, вести учет безопасности данных и т. д.