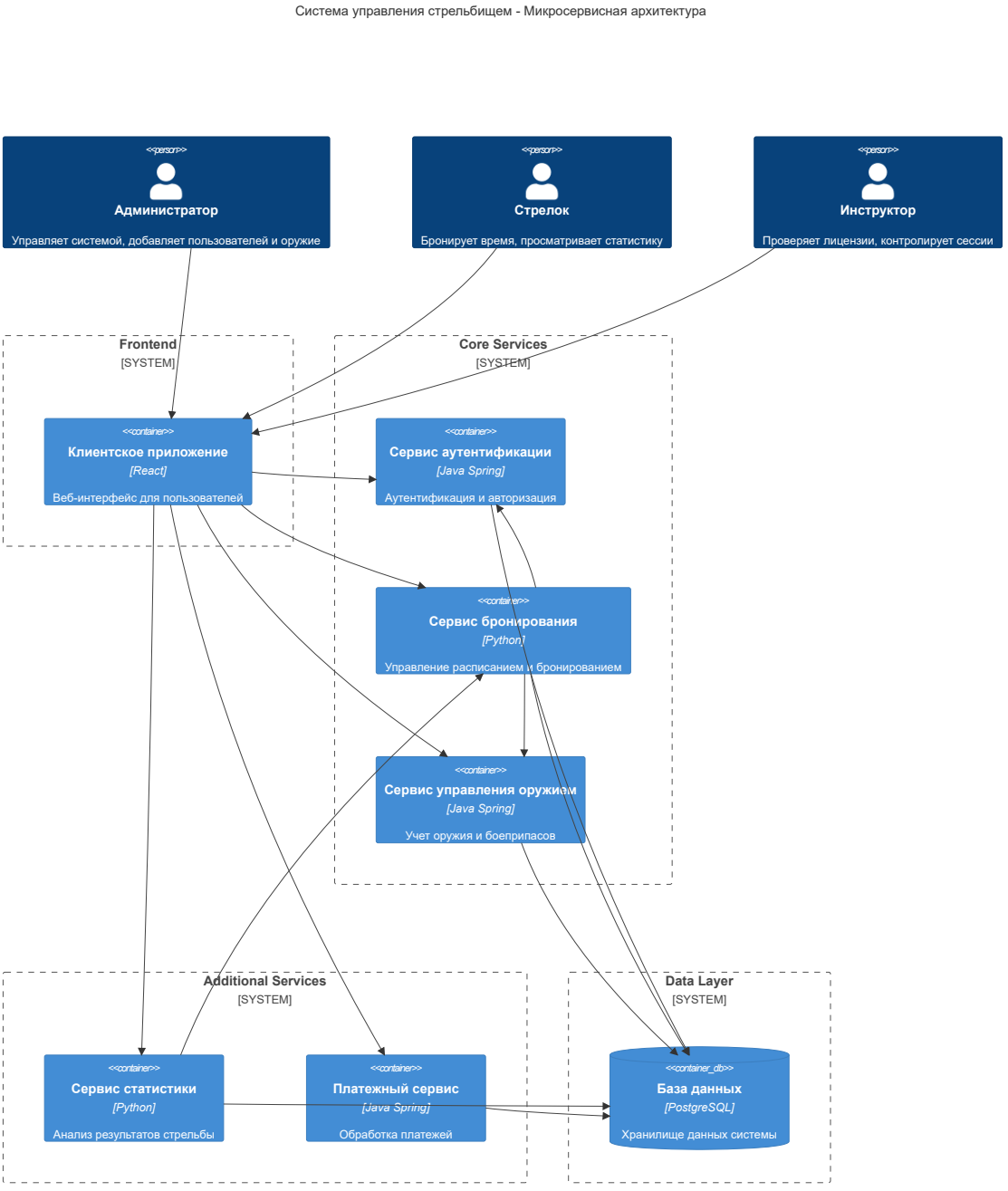
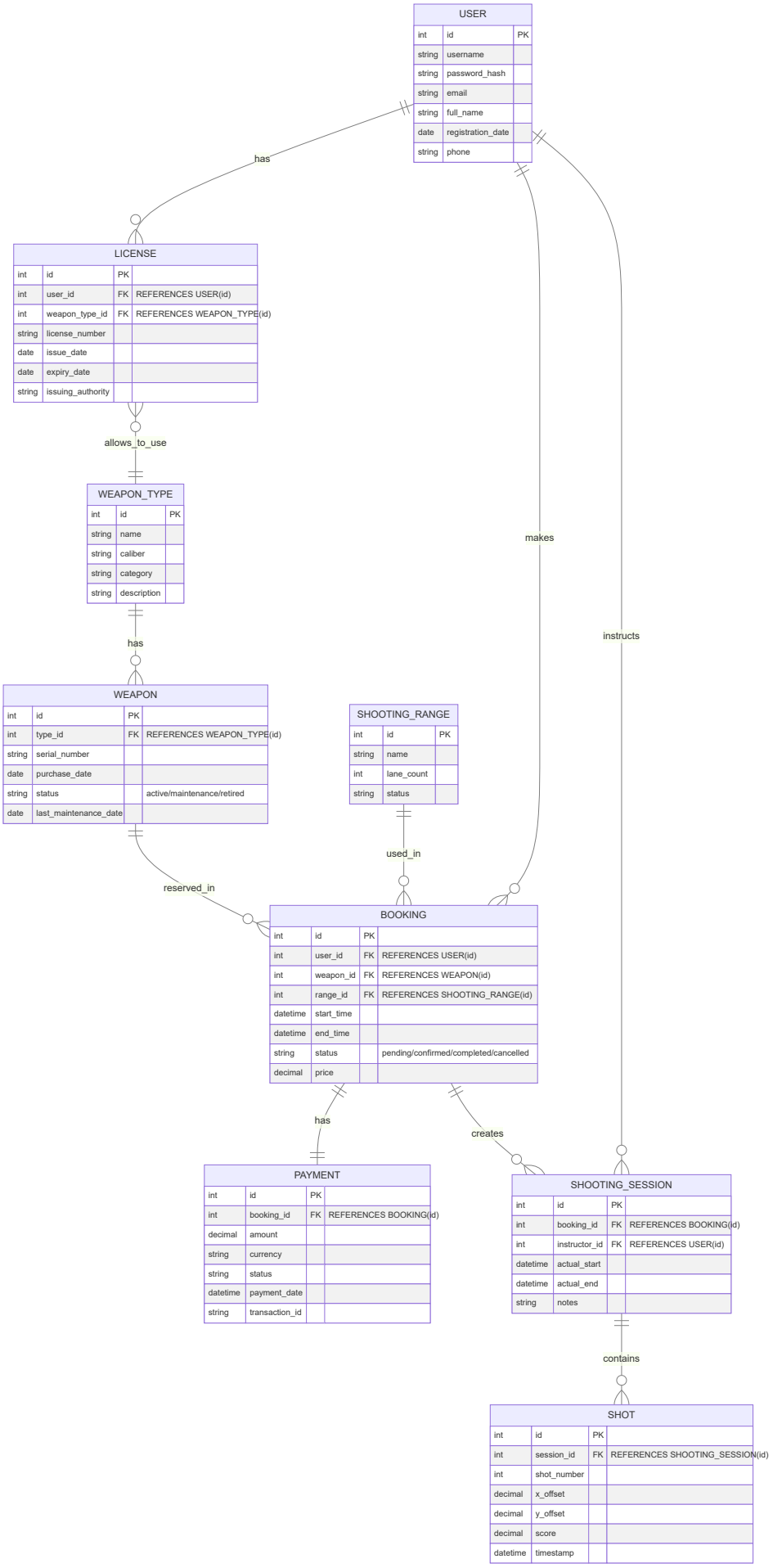
Микросервисная архитектура для системы управления стрельбищем и бронированием огнестрельного оружия

Схема сервисов



P.s Со стрелками косяк вышел, так как писал в формате mermaid, чтобы все блоки красиво были оформлены

Схема базы данных



Описание системы

Разработанная микросервисная архитектура предназначена для автоматизации работы стрелкового клуба, предоставляющего услуги по аренде огнестрельного оружия и организации стрельбы.

1. Процесс бронирования стрелковой сессии:

Когда пользователь (стрелок) хочет забронировать время для стрельбы, происходит следующая последовательность взаимодействий:

1. Клиентское приложение отправляет запрос к **Сервису бронирования** (Python) на доступные временные слоты
2. **Сервис бронирования** запрашивает у **Сервиса аутентификации** (Java Spring) проверку лицензии пользователя
3. **Сервис аутентификации** проверяет в базе данных наличие действующей лицензии и возвращает ответ
4. **Сервис бронирования** запрашивает у **Сервиса управления оружием** (Java Spring) список доступного оружия соответствующих типов
5. После получения всей информации **Сервис бронирования** формирует ответ клиенту с доступными слотами и оружием.
6. Когда пользователь выбирает слот и оружие, клиентское приложение отправляет запрос на создание брони
7. **Сервис бронирования** создает запись в БД и отправляет запрос в **Платежный сервис** (Java Spring) на создание счета:

2. Процесс проведения стрелковой сессии:

1. При прибытии на стрельбище пользователь подтверждает начало сессии через клиентское приложение
2. **Сервис бронирования** отмечает сессию как начатую и уведомляет **Сервис статистики** (Python)
3. Во время стрельбы система трекинга (внешняя) отправляет данные о каждом выстреле в **Сервис статистики**
4. **Сервис статистики** сохраняет данные выстрела и периодически агрегирует промежуточные результаты
5. По окончании сессии инструктор подтверждает завершение через клиентское приложение

3. Процесс анализа результатов:

1. Когда пользователь хочет посмотреть свою статистику, клиентское приложение запрашивает данные
2. **Сервис статистики** запрашивает у **Сервиса бронирования** список сессий за период
3. **Сервис статистики** вычисляет различные метрики (точность, прогресс, сравнение с другими) и возвращает клиенту

Технические детали взаимодействия:

1. Все межсервисные запросы аутентифицируются с помощью JWT-токенов, выданных **Сервисом аутентификации**.
2. Для обеспечения надежности:
   - Retry-механизмы при временных ошибках
   - Circuit breakers для предотвращения каскадных сбоев
   - Асинхронная обработка длительных операций (например, генерации отчетов)
3. Формат данных - JSON для всех запросов и ответов.
4. Каждый сервис имеет свой набор API-эндпоинтов и обрабатывает ошибки стандартизированным образом
5. Для событий, не требующих немедленной обработки (например, запись статистики выстрелов), используется очередь сообщений (на базе PostgreSQL).