**Отчет о системе продажи билетов на спортивные мероприятия (TicketBookingSystem)**

**Архитектура приложения**

Приложение разработано с использованием трехслойной архитектуры, которая включает следующие слои:

1. **Слой представления данных**:
   * Этот слой отвечает за взаимодействие с пользователем и отображение данных. Используются JSP страницы для рендеринга пользовательского интерфейса.
   * Пользовательские запросы поступают через сервлеты, которые обрабатывают запросы и вызывают соответствующие методы сервисного слоя.
2. **Service-слой**:
   * Этот слой содержит бизнес-логику приложения. Он взаимодействует с DAO (Data Access Object) для выполнения операций с базой данных.
   * Сервисные классы отвечают за выполнение операций, таких как создание, обновление и удаление сущностей, а также за проведение валидации данных.
3. **Слой доступа к данным (DAO)**:
   * Этот слой отвечает за взаимодействие с базой данных. Используется Hibernate.
   * DAO классы выполняют CRUD (Create, Read, Update, Delete) операции и выполняют запросы к базе данных.

**Использованные технологии**

1. **Java EE**:
   * Основной платформой для разработки приложения является Java Enterprise Edition (Java EE), которая обеспечивает стандарты и библиотеки для создания корпоративных приложений.
2. **Hibernate**:
   * Hibernate используется как ORM-фреймворк для управления объектно-реляционными отображениями и взаимодействия с базой данных PostgreSQL.
3. **PostgreSQL**:
   * В качестве системы управления базами данных используется PostgreSQL.
4. **JSP**:
   * JavaServer Pages (JSP) используются для создания динамических веб-страниц, которые взаимодействуют с пользователем.

**Особенности реализации**

1. **Фильтры**:
   * Реализованы фильтры LocaleFilter и AuthorizationFilter для управления локализацией и авторизацией пользователей.
   * LocaleFilter обрабатывает изменение языка интерфейса на основе параметров URL и cookie.
   * AuthorizationFilter проверяет права доступа пользователей в зависимости от их роли и запрашиваемого URL.
2. **Валидация данных**:
   * Валидация данных осуществляется на уровне сервисного слоя с использованием валидаторов, таких как CreateOrUpdateArenaValidator.
   * Это обеспечивает проверку корректности данных перед их сохранением или обновлением в базе данных.
3. **DTO и Мапперы**:
   * Используются DTO (Data Transfer Objects) для передачи данных между слоями приложения.
   * Мапперы преобразуют сущности в DTO и наоборот, обеспечивая изоляцию слоев и упрощение передачи данных.
4. **Управление транзакциями**:
   * Управление транзакциями осуществляется с использованием Hibernate, что обеспечивает целостность данных и корректность выполнения операций с базой данных.
5. **Юнит-тестирование**:
   * На данный момент реализованы юнит-тесты для DAO классов, что позволяет проверить корректность взаимодействия с базой данных (для тестов используется H2 база).