**Аутентификация/авторизация/регистрация**

* GET /login – получить форму авторизации (возвращает страничку *login.jsp*);
* GET /registration – получить форму регистрации (возвращает страничку *registration.jsp*);
* POST /registration – создать нового пользователя в системе;
* POST /logout – выйти из учетной записи (возврат к форме авторизации);
* Отдельного метода контроллера под POST-запрос при логине нет, так как аутентификация выполнена с помощью Spring Security (настроен собственный класс CustomUserDetailsService).

**Функциональность администратора**

* GET /admin – переход в главное меню администратора (возвращает *admin\_panel.jsp*).

API для работы с сущностью *Arena* (отражает спортивные арены для мероприятий, на которые пользователям предлагается купить билеты):

* GET /admin/arenas – получить список арен;
* POST /admin/arenas/create – создать новую арену;
* POST /admin/arenas/{id}/update – обновить информацию об арене;
* POST /admin/arenas/{id}/delete – удалить информацию об арене.

API для работы с сущностью *Sector* (отражает сектора спортивных арен, содержащие в себе ряды мест для покупки билетов):

* GET /admin/sectors?arenaId={arenaId} – получить список секторов конкретной арены;
* POST /admin/sectors/create?arenaId={arenaId} – создать новый сектор;
* POST /admin/sectors/{id}/update?arenaId={arenaId} – обновить информацию о секторе;
* POST admin/sectors/{id}/delete – удалить информацию о секторе.

API для работы с сущностью *Row* (отражает ряды секторов арен):

* GET admin/rows?arenaId={arenaId}&sectorId={sectorId} – получить список рядов для конкретного сектора конкретной арены;
* POST admin/rows/create?arenaId={arenaId}&sectorId={sectorId} – создать новый ряд;
* POST /admin/rows/{id}/update?sectorId={sectorId}&arenaId={arenaId} – обновить информацию о ряде;
* POST admin/rows/{id}/delete – удалить информацию о ряде.

API для работы с сущностью *SportEvent* (отражает доступные спортивные события/мероприятия для покупки билетов пользователями):

* GET /admin/sport\_events – получить список спортивных мероприятий;
* POST admin/sport\_events/create – создать новое событие;
* POST /admin/sport\_events/{id}/update – обновить информацию о спортивном событии;
* POST /admin/sport\_events/{id}/delete – удалить информацию о спортивном событии.

API для работы с сущностью *Ticket* (билеты на спортивные события):

* GET /admin/tickets?eventId={eventId}– получить список доступных билетов на конкретное спортивное событие;
* POST /admin/tickets/create?eventId={eventId}– создать новый билет;
* POST /admin/tickets/{id}/update?eventId={eventId}&seatId={seatId} – обновить информацию о билете;
* POST /admin/tickets/{id}/delete?eventId={eventId}&seatId={seatId} –

удалить информацию о билете.

**Функциональность пользователя**

* GET /view\_available\_events – получить список доступных спортивных мероприятий для бронирования билетов;
* GET /view\_available\_tickets?id={id} – получить список доступных билетов для бронирования на конкретное спортивное мероприятие;
* GET /user\_cart – получить список добавленных в корзину билетов пользователем
* POST /user\_cart?ticketId={ticketId}&action={action} – добавить элемент в корзину/удалить элемент из корзины/очистить всю корзину (зависит от значения передаваемого параметра *action*: *add*, *remove*, *clear*);
* GET /purchase – получить форму оформления покупки билетов;
* POST /purchase – осуществить покупку выбранных билетов, указав реквизиты карты;
* GET /purchasedTickets – просмотр купленных билетов пользователем (с возможностью сохранить в формате pdf)