Задание **2**
Описание API 2
GET /game/:id 2
Ответ 2
Возможные коды ответа 3
GET /game?limit=:limit&offset=:offset 3
Ответ 3
Возможные коды ответа 3
POST /game/:id 4
Запрос 4
Ответ 4
Возможные коды ответа 5
POST /game 5
Запрос 5
Проверки данных 6
Ответ 6
Возможные коды ответа 7
POST /user/login 7
Запрос 7
Ответ 7
Возможные коды ответа 7
POST /user/logout 8
Запрос 8
Возможные коды ответа 8
POST /user 8
Запрос 8
Проверки данных 9
Возможные коды ответа 9
GET /user/:username 9
Ответ 9
Возможные коды ответа 9
GET /user?limit=:limit&offset=:offset 10
Ответ 10
Возможные коды ответа 10
POST /debug/reset 10
Запрос 10
Возможные коды ответа 10

Задание

Реализовать бекенд для сетевой пошаговой игры в крестики-нолики на Scala.

В своей реализации для хранения игр, шагов, пользователей и сессий надо использовать in-memory реляционную базу данных h2 или in-memory реализацию mongo - fongo с любой удобной библиотекой для работы с ними. Как Web сервер надо выбрать Play или akka-http. Обрабатывать запросы пользователей и в базу можно как синхронно, так и асинхронно, используя, например, Future. Архитектуру приложения нужно разбить на уровни (api/логика игры/хранилище) и написать для каждого уровня тесты.

В приложении пользователи для авторизации используют заголовок сессии “session”ю Время жизни сессии должно быть 5 минут и при каждом запросе время жизни должно сбрасываться опять на 5 минут, то есть сессия будет считаться истекшей через 5 минут бездействия (отсутствия запросов) от пользователя.

До последнего занятия (29 мая) мне нужно прислать репозиторий на github или bitbucket с готовым к запуску приложением, которое я запущу и буду проверять. Приложение должно собираться с помощью sbt, запускаться командой “sbt run” и слушать http запросы на 8080 порту.

Описание API

GET /game/:id

Возвращает данные игры по id.

Ответ

```
{
	"id": 7,
	"next_step": "vasya",
	"won": null,
	"finished": false,
	"players": ["vasya", "nagibator"],
	"steps": 6,
	"field": [
		[0, 0, 2, 0, 2],
		[0, 0, 1, 1, 0],
		[0, 0, 1, 2, 0],
```

```
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0]
        ]
}
```

Возможные коды ответа

200 (Ок)	
404 (Not Found)	Если игра с таким id не найдена

GET /game?limit=:limit&offset=:offset

Возвращает все данные по играм, отсортированные по id. Количество возвращаемых данных должно быть ограничено GET параметром limit, а сдвиг - с какого по порядку элемента возвращать элементы - в параметре offset.

Ответ

```
[{
        "id": 7,
        "next_step": "vasya",
        "won": null,
        "finished": false,
        "players": ["vasya", "nagibator"],
        "steps": 6,
        "field": [
                [0, 0, 2, 0, 2],
                [0, 0, 1, 1, 0],
                [0, 0, 1, 2, 0],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0]
        ]
}, {...}, {...}]
```

Возможные коды ответа

200 (Ок)	

400 (Bad Request)	Если limit <= 0 или offset < 0

POST /game/:id

Делает шаг в игре с id. Только аутентифицированный участник игры может сделать это.

Запрос

```
Header "session: uid"
{
        "step": [0, 1]
}
```

Массив “step” является массивом координат [x, y], например [0, 1] соответствует выделенной ячейке

	x0	x1	x2
y0	0	1	2
y1	**0**	0	1
y2	0	2	2

Клетка 0 - пустая, клетки 1 и 2 соответствуют крестикам первого и второго игроков в списке "players" в свойствах игры.

Ответ

```
{
        "id": 1,
        "next_step": "nagibator",
        "won": null,
        "finished": false,
        "players": ["vasya", "nagibator"],
        "steps": 7,
        "size": [5, 5],
        "crosses_length_to_win": 3,
        "field": [
```

```
                [0, 0, 2, 0, 2],
                [1, 0, 1, 1, 0],
                [0, 0, 1, 2, 0],
                [0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0]
        ]
}
```

Возможные коды ответа

200 (Ок)	Если шаг выполнен
401 (Unauthorized)	Если нет заголовка session, он невалидный или сессия истекла
403 (Forbidden)	Если у пользователя нет права ходить в этой игре
404 (Not Found)	Если игра с таким id не найдена
400 (Bad Request)	Если поле меньше координат шага, если игра закончена, если чужая очередь хода или данные не соответствуют спецификации
409 (Conflict)	Если это поле уже занято

POST /game

Создает новую игру. Создать новую игру может только аутентифицированный пользователь. Только аутентифицированный пользователь может сделать это.

Запрос

```
Header "session: uid"
{
        "opponent": "vasya",
        "size": [3, 3],
        "first_step_by": "vasya",
        "crosses_length_to_win": 3
```

```
}
```

Проверки данных

“opponent” существует
Каждое измерение поля “size” >= 3 и < 10
“crosses_length_to_win” - сколько нужно крестиков в ряд для победы. Должно быть <= min(size(0), size(1)) и >= 3.
Первый игрок, который ходит - “first_step_by” существует и является создающим или его оппонентом

Ответ

```
{
	"id": 7,
	"next_step": "vasya",
	"won": null,
	"finished": false,
	"players": ["vasya", "nagibator"],
	"steps": 0,
	"size": [3, 3],
	"crosses_length_to_win": 3,
	"field": [
		[0, 0, 0],
		[0, 0, 0],
		[0, 0, 0],
	]
}
```

Игра заканчивается когда в ряд по горизонтали, вертикали или диагонали будет выстроено в ряд “crosses_length_to_win” крестиков одного из игроков, он и будет являться победителем, или если всё поле заполнено, а условие не выполнено, то победителя нет ("won": null) - ничья. В любом случае поле “finished” должно стать true, а "next_step" - null. Поле может быть прямоугольным.
В списке “players” создающий игрок должен быть первым, а его оппонент - вторым. Соответственно на поле их крестики будут заняты цифрами 1 и 2 - по номеру в этом списке.

Возможные коды ответа

200 (Ок)	
401 (Unauthorized)	Если нет заголовка session, он невалидный или сессия истекла
400 (Bad Request)	Если были переданные неверные данные игры

POST /user/login

Авторизация зарегистрированного пользователя, создает UID идентификатор сессии для использования в других запросах.

Запрос

```
{
        "username": "vasya",
        "password": "c4Gf4g4g"
}
```

Ответ

```
{
        "session": "89ca0f61-75a2-4a4f-b495-d11a62d01804"
}
```

Возвращаемый код сессии должен передаваться в другие запросы в заголовке “session” для аутентификации пользователя. У сессии должен быть срок жизни 5 минут, после чего сессию нужно открывать заново и из хранилища сессий она должен быть удалена.

Возможные коды ответа

200 (Ок)	Если у пользователя уже была сессия, то можно вернуть её же

403 (Forbidden)	Если логин и пароль неверны
400 (Bad Request)	Если данные не соответствуют спецификации

POST /user/logout

Закрывает открытую сессию пользователя.

Запрос

Header "session: uid"

Возможные коды ответа

200 (Ок)	Если сессия была активна и была закрыта
403 (Forbidden)	Если закрываемая сессия принадлежит другому пользователю
401 (Unauthorized)	Пользователь не авторизован или сессия устарела (должна быть удалена автоматически)
400 (Bad Request)	Если заголовок отсутствует

POST /user

Регистрация нового пользователя.

Запрос

```
{
	"username": "vasya",
	"password": "c4Gf4g4g"
}
```

Проверки данных

“username” от 3 до 20 символов
"password" от 8 до 100 символов

В базе нельзя хранить пароль в простом виде, нужно хранить хеш от пароля! Например, sha3-256. Для проверки нужно захешировать пришедший пароль и сравнить с хешем в базе.

Возможные коды ответа

200 (Ок)	Пользователь зарегистрирован
409 (Conflict)	Если такой username уже зарегистрирован
400 (Bad Request)	Если данные не соответствуют спецификации

GET /user/:username

Возвращает информацию по пользователю.

Ответ

```
{
        "username": "vasya",
        "online": true,
        "wins": 0,
        "losses": 0
}
```

Поле "online" должно быть true, если у пользователя есть активная сессия. Поля "wins" и "losses" - статистика побед и поражений пользователя.

Возможные коды ответа

200 (Ок)	

404 (Not Found)	Если такой username не найден

GET /user?limit=:limit&offset=:offset

Возвращает всех пользователей в порядке регистрации. Количество возвращаемых данных должно быть ограничено GET параметром limit, а сдвиг - с какого по порядку элемента возвращать элементы - в параметре offset.

Ответ

```
[{
        "username": "vasya",
        "online": true,
        "wins": 0,
        "losses": 0
}, {...}, {...}]
```

Возможные коды ответа

200 (Ок)	
400 (Bad Request)	Если limit <= 0 или offset < 0

POST /debug/reset

Удаляет все данные из базы данных. Только суперпользователь может сделать это - должен быть установлен заголовок “admin” с любым значением.

Запрос

Header "admin: admin"

Возможные коды ответа

200 (Ок)	

401 (Unauthorized)	Если нет заголовка admin