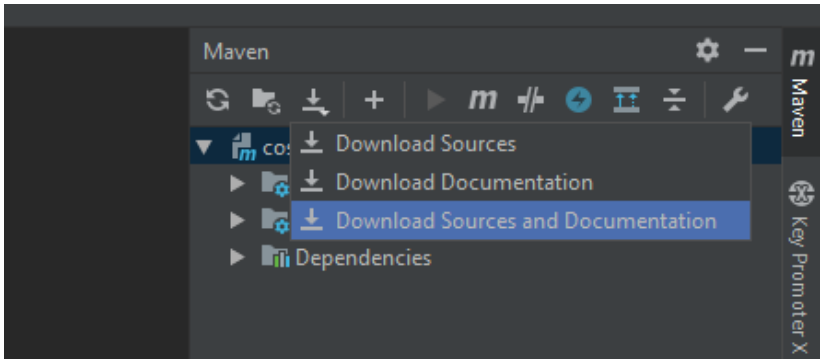
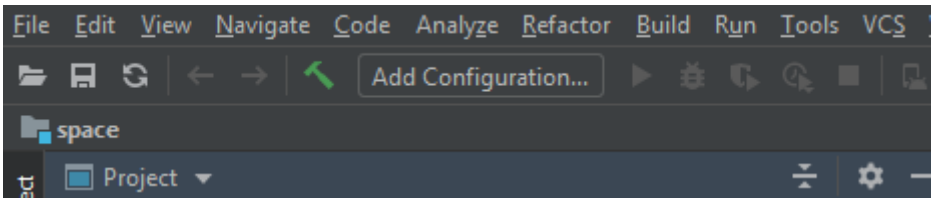
Как попасть на стажировку?

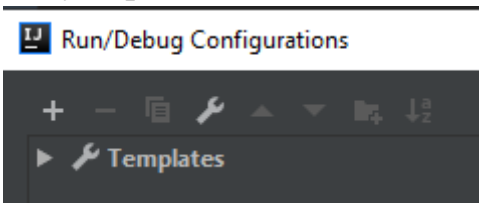
1. Скачать Intellij IDEA Ultimate Edition. Ее пробный период — 30 дней, которых хватит на решение тестового задания. А на стажировке ты получишь ключ для доступа на протяжении 6 месяцев.
2. Открыть проект, используя файл pom.xml. Так он сразу станет Maven-проектом. Maven встроен в Intellij IDEA Ultimate Edition, но также можно скачать и установить его отдельно.
3. Справа во вкладке Maven нажать на Download Sources and Documentation для загрузки исходного кода и документации используемых фреймворков.



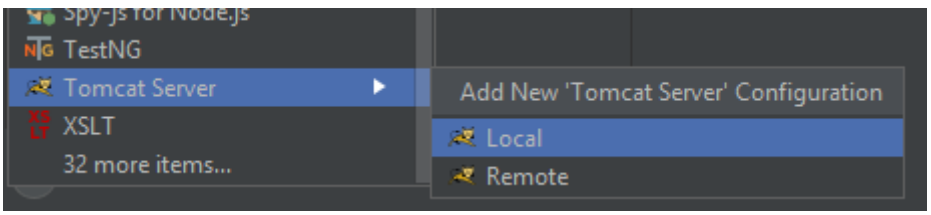
4. Установить MySQL сервер. Логин и пароль **root**. Порт **3306**. Залогиниться и выполнить скрипт init.sql, который ты найдешь в корне проекта.
5. Собрать проект (в терминале перейти в папку с pom файлом и выполнить: mvn -DskipTests=true clean install) (чтобы Maven работал в терминале, он должен быть прописан в системной переменной PATH)
6. Настроить запуск приложения через Tomcat (если не установлен, то можно скачать отсюда):
   6.1 Выбрать Add Configuration... в параметрах запуска:



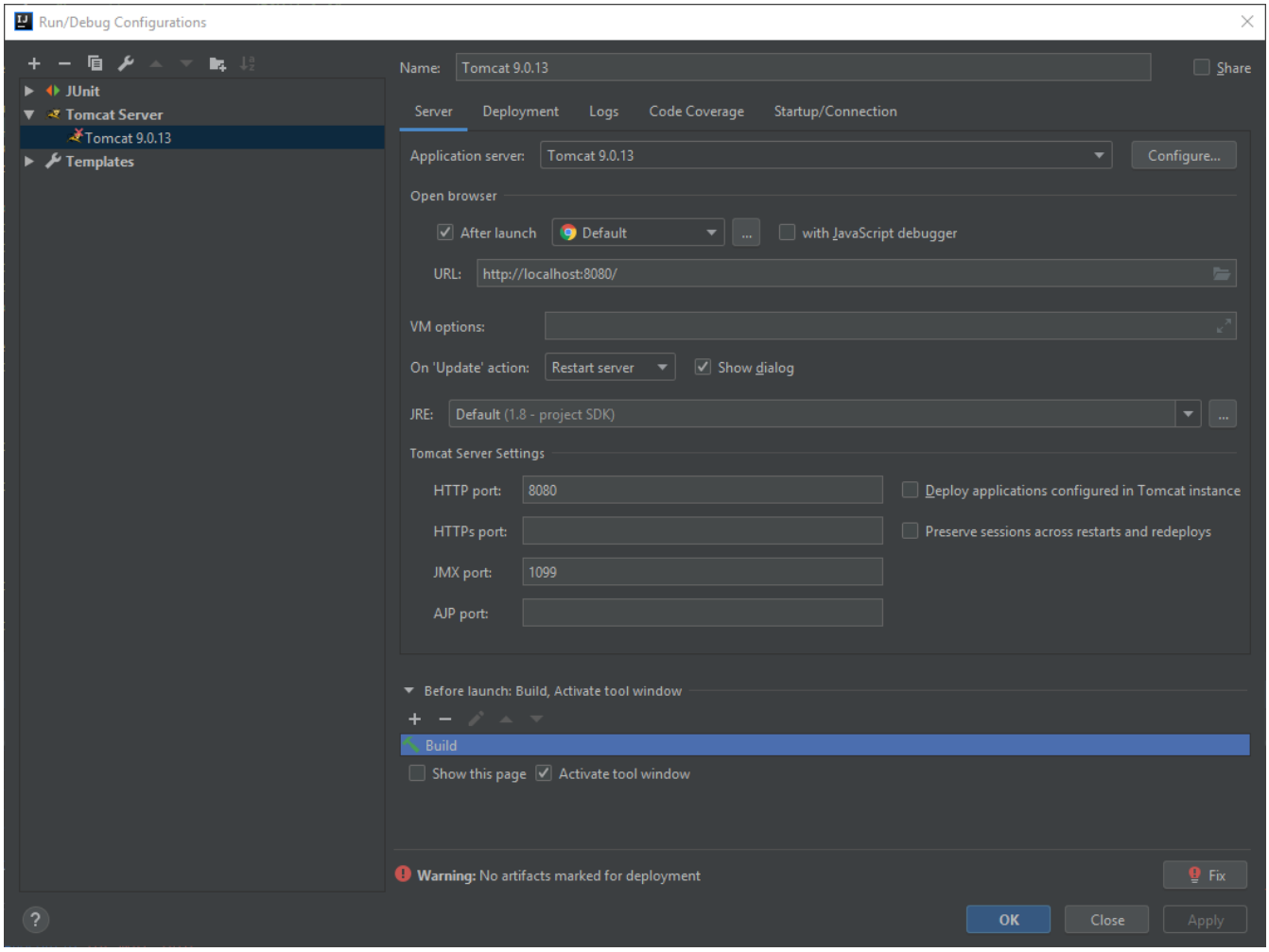
   6.2 Нажать на «плюсик» чтобы раскрыть список возможных конфигураций. Tomcat может находиться в самом низу. При необходимости нажми на «more items».



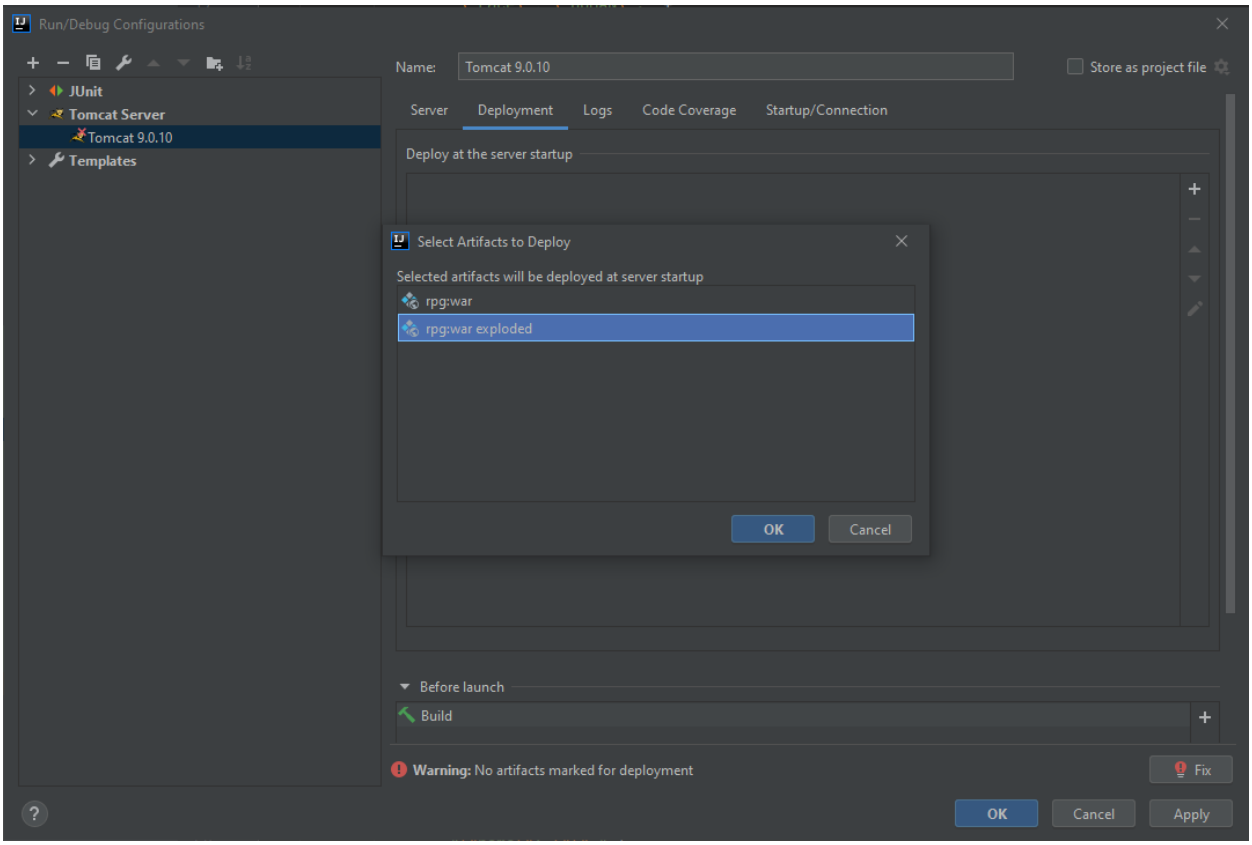
   6.3 В выпадающем меню выбрать Local.



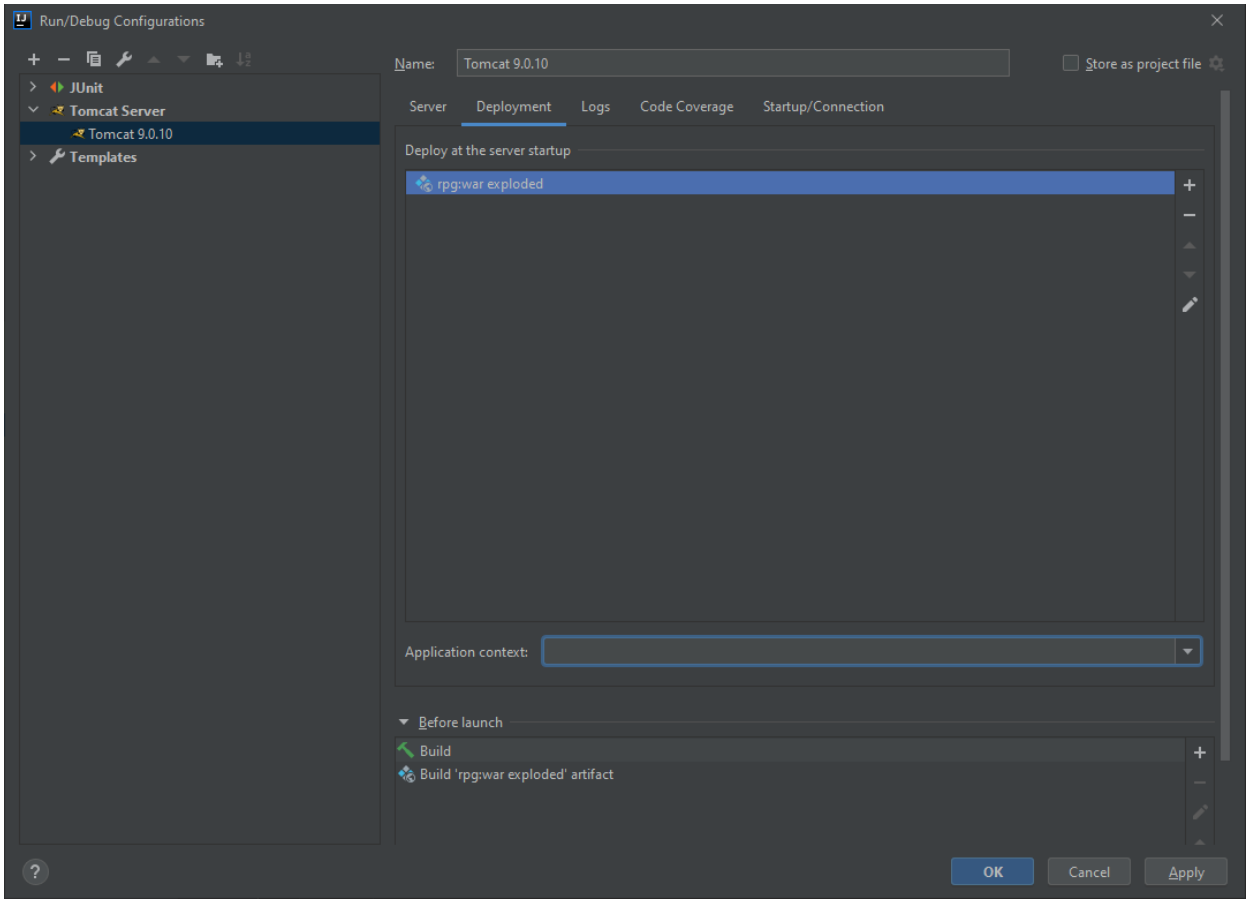
6.4 Во вкладке Server настроить все так, как показано на скриншоте (используй версию Tomcat 9):



6.5 Во вкладке Deployment нажать на «плюсик» и добавить артефакт «rpg:war exploded», как указано на скриншоте.



6.6 Очистить поле Application context и нажать Ок:



7. Запустить приложение. В браузере откроется стартовая страница, на которой ты увидишь готовый интерфейс приложения, но он не работает пока на сервере нет соответствующего функционала. По мере реализации проекта, интерфейс будет правильно отображать данные и отправлять запросы на редактирование, удаление и создание игроков.
8. Дописать нужный функционал (см. Задание на стажировку). **ВАЖНО: Файл pom.xml изменять нельзя!**
9. Протестировать свое приложение с помощью тестов, которые уже есть в проекте.
10. Если **ВСЕ ТЕСТЫ ПРОШЛИ** успешно, то отправить задание на проверку в JavaRush.

Задание на стажировку.

Нужно дописать приложение для администратора сетевой ролевой игры, где он сможет редактировать параметры персонажей (игроков), и раздавать баны. Должны быть реализованы следующие возможности:

1. получать список всех зарегистрированных игроков;
2. создавать нового игрока;
3. редактировать характеристики существующего игрока;
4. удалять игрока;
5. получать игрока по id;
6. получать отфильтрованный список игроков в соответствии с переданными фильтрами;
7. получать количество игроков, которые соответствуют фильтрам.

Для этого необходимо реализовать REST API в соответствии с документацией.

В проекте должна использоваться сущность Player, которая имеет поля:

Long id	ID игрока
String name	Имя персонажа (до 12 знаков включительно)
String title	Титул персонажа (до 30 знаков включительно)
Race race	Расса персонажа
Profession profession	Профессия персонажа
Integer experience	Опыт персонажа. Диапазон значений 0..10,000,000
Integer level	Уровень персонажа
Integer untilNextLevel	Остаток опыта до следующего уровня
Date birthday	Дата регистрации Диапазон значений года 2000..3000 включительно
Boolean banned	Забанен / не забанен

Также должна присутствовать бизнес-логика:
Перед сохранением персонажа в базу данных (при добавлении нового или при апдейте характеристик существующего), должны высчитываться:
- текущий уровень персонажа
- опыт необходимый для достижения следующего уровня
и сохраняться в БД. Текущий уровень персонажа рассчитывается по формуле:

$$L = \frac{\sqrt{2500 + 200 \cdot \exp} - 50}{100},$$

где:
exp — опыт персонажа.

Опыт до следующего уровня рассчитывается по формуле:

$$N = 50 \cdot (lvl + 1) \cdot (lvl + 2) - exp,$$

где:
lvl — текущий уровень персонажа;
exp — опыт персонажа.

В приложении используй технологии:

1. Maven (для сборки проекта);
2. Tomcat 9 (для запуска своего приложения);
3. Spring;
4. Spring Data JPA;
5. MySQL (база данных (БД)).

Вот ссылка на архив с полезными книгами, которые помогут тебе в решении тестового задания. Не нужно их все перечитать, используй как справочники.

Отправка готового задания на проверку.

Результат нужно выложить на GitHub в публичный репозиторий.
Перед этим убедись, что все тесты проходят. **ВАЖНО: Файл pom.xml изменять нельзя!**
На странице стажировки нажми кнопку «Отправить заявку», заполни все поля и нажми «Отправить заявку». Дождись результатов автоматической проверки задания (этот процесс занимает до 1 мин). В отдельных случаях может понадобиться ручная проверка задания, которая может длиться несколько дней.

Обрати внимание.

1. Если в запросе на создание игрока нет параметра “banned”, то считаем, что пришло значение “false”.
2. Параметры даты между фронтом и сервером передаются в миллисекундах (тип Long) начиная с 01.01.1970.
3. При обновлении или создании игрока игнорируем параметры “id”, “level” и “untilNextLevel” из тела запроса.
4. Если параметр order не указан – нужно использовать значение PlayerOrder.ID.
5. Если параметр pageNumber не указан – нужно использовать значение 0.
6. Если параметр pageSize не указан – нужно использовать значение 3.
7. Не валидным считается id, если он:
    - не числовой
    - не целое число
    - не положительный
8. При передаче границ диапазонов (параметры с именами, которые начинаются на «min» или «max») границы нужно использовать включительно.

REST API.

Get players list

URL	/rest/players
Method	GET
URL Params	**Optional:** String name, String title, Race race, Profession profession, Long after, Long before, Boolean banned, Integer minExperience, Integer maxExperience, Integer minLevel, Integer maxLevel, PlayerOrder order, Integer pageNumber, Integer pageSize
Data Params	None
Success Response	**Code:** 200 OK **Content:** [{ “id”:[Long], “name”:[String], “title”:[String], “race”:[Race], “profession”:[Profession], “birthday”:[Long], “banned”:[Boolean], “experience”:[Integer], “level”:[Integer], “untilNextLevel”:[Integer] }, …]
Notes	Поиск по полям name и title происходить по частичному соответствию. Например, если в БД есть игрок с именем «Камираж», а параметр name задан как «ир» - такой игрок должен отображаться в результатах (Кам**ир**аж). pageNumber – параметр, который отвечает за номер отображаемой страницы при использовании пейджинга. Нумерация начинается с нуля pageSize – параметр, который отвечает за количество результатов на одной странице при пейджинге

Get players count

URL	/rest/players/count
Method	GET
URL Params	**Optional:** String name, String title, Race race, Profession profession, Long after, Long before, Boolean banned, Integer minExperience, Integer maxExperience, Integer minLevel, Integer maxLevel
Data Params	None
Success Response	**Code:** 200 OK **Content:** Integer
Notes	

Create player

URL	/rest/players
Method	POST
URL Params	None

Data Params	{
	“name”:[String],
	“title”:[String],
	“race”:[Race],
	“profession”:[Profession],
	“birthday”:[Long],
	“banned”:[Boolean], --optional, default=false
	“experience”:[Integer]
	}

Success Response	**Code:** 200 OK
	Content: {
	“id”:[Long],
	“name”:[String],
	“title”:[String],
	“race”:[Race],
	“profession”:[Profession],
	“birthday”:[Long],
	“banned”:[Boolean],
	“experience”:[Integer],
	“level”:[Integer],
	“untilNextLevel”:[Integer]
	}

Notes	Мы не можем создать игрока, если: - указаны не все параметры из **Data Params (**кроме banned**)**; - длина значения параметра “name” или “title” превышает размер соответствующего поля в БД (12 и 30 символов); - значение параметра “name” пустая строка; - опыт находится вне заданных пределов; - “birthday”:[Long] < 0; - дата регистрации находятся вне заданных пределов. В случае всего вышеперечисленного необходимо ответить ошибкой с кодом 400.

Get player

URL	/rest/players/{id}
Method	GET
URL Params	id
Data Params	None

Success Response	**Code:** 200 OK
	Content: {
	“id”:[Long],
	“name”:[String],
	“title”:[String],
	“race”:[Race],
	“profession”:[Profession],
	“birthday”:[Long],
	“banned”:[Boolean],
	“experience”:[Integer],
	“level”:[Integer],
	“untilNextLevel”:[Integer]
	}

Notes	Если игрок не найден в БД, необходимо ответить ошибкой с кодом 404. Если значение id не валидное, необходимо ответить ошибкой с кодом 400.

Update player

URL	/rest/players/{id}
Method	POST
URL Params	id

Data Params	{
	“name”:[String], --optional
	“title”:[String], --optional
	“race”:[Race], --optional
	“profession”:[Profession], --optional
	“birthday”:[Long], --optional
	“banned”:[Boolean], --optional
	“experience”:[Integer] --optional
	}

Success Response	**Code:** 200 OK
	Content: {
	“id”:[Long],
	“name”:[String],
	“title”:[String],
	“race”:[Race],
	“profession”:[Profession],
	“birthday”:[Long],
	“banned”:[Boolean],
	“experience”:[Integer],
	“level”:[Integer],
	“untilNextLevel”:[Integer]
	}

Notes	Обновлять нужно только те поля, которые не null. Если игрок не найден в БД, необходимо ответить ошибкой с кодом 404. Если значение id не валидное, необходимо ответить ошибкой с кодом 400.

Delete player

URL	/rest/players/{id}
Method	DELETE
URL Params	id
Data Params	
Success Response	**Code:** 200 OK
Notes	Если игрок не найден в БД, необходимо ответить ошибкой с кодом 404. Если значение id не валидное, необходимо ответить ошибкой с кодом 400.