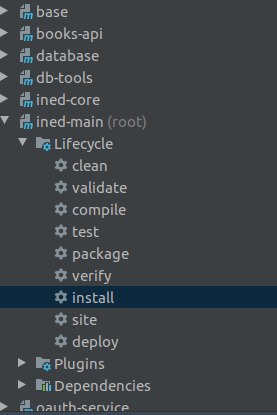
# Инструкция по использованию

1. Заранее извиняюсь, что использовал бд PostgreSQL
2. Для начала необходимо запустить базу данных. Сделать можно 2 разными способами.
   1. Первый способ это подготовленный мною докер-композ файл. Нужно перейти в папку ined-main/docker и выполнить следующий ряд команд:
      1. docker-compose up -d запустится контейнер с созданной базой данных в фоновом режиме.
      2. docker-compose ps проверить все ли запустилось.
   2. Если не установлен docker, то можно воспользоваться локальной Postgresql и в обязательном порядке создать базу данных с названием «ined\_db»
3. Импортировать maven проект выбрав в папке «ined-main/» главный pom.xml
4. Далее необходимо собрать maven проект. В правой стороне меню IJ открыть maven. Далее выбрать главный maven модуль ined-main(root), перейти в Lifecycle и запустить поочереди clean и install(Как на рисунке ниже). С помощью данных операций сгенерируются jar файлы, так же будут созданы сущности в базе данных.



1. Чтобы загрузить тестовые данные в базу данных, в папке TestSQL, где вы нашли данную инструкцию, имеется sql скрипт для создание тестовых данных.
2. Запустить BooksApiApplication. Это все для бэкенда.

## Варианты проверки

С помощью отправителя запросов( условный Postman). Все запросы приходятся на локальный сервер открытый на 8084 порту(localhost:8084/).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Api | Method | Headers | Body | Return | Description |
| 1 | user/api/newUser/ | POST | - | email:String  password: String  firstName: String  lastName: String  roleID: Long | HttpStatus.OK | Регистрация пользователя, уникальный email. |
| 2 | /api/auth/login/ | POST | - | email:String  password:String | access\_token | Аутентификация и авторизация и создание токена. |
| 3 | /main/api/books/ | POST | Authorization: Bearer token(optional) | - | BooksList | Получение всех возможных книг |
| 4 | /main/api/bookByID | POST | Authorization: Bearer token(optional) | id:Long | Book | Получение одной книги |
| 5 | user/api/comment/newComment | POST | Authorization: Bearer token | id:Long  commentText:String  email:String  bookID:Long | new Comment | Создание комментария |
| 6 | user/api/comment/deleteComment | POST | Authorization: Bearer token | id:Long | HttpStatus.OK | Удаление комментария |
| 7 | admin/api/addBook | POST | Authorization: Bearer token | name:String  bookStyle:BookStyles  authors:List<Authors> | Book | Добавление новой книги |
| 8 | admin/api/updateBook | POST | Authorization: Bearer token | id:Long  name:String  bookStyle:BookStyles  authors:List<Authors> | Book | Обновление книги |
| 9 | admin/api/deleteBook | POST | Authorization: Bearer token | id:Long | HttpStatus.Ok | Удаление книги |
| 10 | admin/api/getAllBooks | POST | Authorization: Bearer token | - | BooksList | Получение всех книг, для будущих манипуляций |
| 11 | admin/api/addAuthor | POST | Authorization: Bearer token | firstName:String  lastName:String | Author | Добавление нового автора |
| 12 | admin/api/updateAuthor | POST | Authorization: Bearer token | id:Long  firstName:String  lastName:String | Author | Обновление автора |
| 13 | admin/api/deleteAuthor | POST | Authorization: Bearer token | id:Long | HttpStatus.Ok | Удаление автора |
| 14 | admin/api/getAllAuthors | POST | Authorization: Bearer token | - | BooksList | Получение всех авторов, для будущих манипуляций |
| 15 | admin/api/addBookStyle | POST | Authorization: Bearer token | name:String | BookStyle | Добавление нового жанра |
| 16 | admin/api/updateBookStyle | POST | Authorization: Bearer token | id:Long  name:String | BookStyle | Обновление жанра |
| 17 | admin/api/deleteBookStyle | POST | Authorization: Bearer token | id:Long | HttpStatus.Ok | Удаление жанра |
| 18 | admin/api/getAllBookStyles | POST | Authorization: Bearer token | - | BookStylesList | Получение всех жанров книг, для будущих манипуляций |