Содержание

[1 Вступление 2](#_Toc452318514)

[2 Java 3](#_Toc452318515)

[3 WildFly 3](#_Toc452318516)

[3.1 Ошибка при запуске 3](#_Toc452318517)

[3.2 Добавление пользователя 5](#_Toc452318518)

[4 Среда разработки 6](#_Toc452318519)

[5 Создание проекта в IntelliJ Idea 7](#_Toc452318520)

[5.1 Создание модулей 7](#_Toc452318521)

[5.2 Создание артефакта 13](#_Toc452318522)

[5.3 Создание конфигурации запуска 15](#_Toc452318523)

[5.4 Добавление библиотеки 20](#_Toc452318524)

[5.5 Запуск 22](#_Toc452318525)

[6 Структура проекта 22](#_Toc452318526)

[6.1 Контроллеры 24](#_Toc452318527)

[6.2 Модели 25](#_Toc452318528)

[7 Билд 27](#_Toc452318529)

[8 Работа с WildFly 28](#_Toc452318530)

[8.1 Деплой на сервер приложений 28](#_Toc452318531)

[8.2 Перезагрузка приложения 30](#_Toc452318532)

1. Вступление

Репозиторий инсталляционных пакетов разработан на языке Java с использованием Spring framework 4. Руководство ориентировано на разработку в ОС Windows.

Реализованные функции:

Для пользователей:

* Сквозная аутентификация через Active Directory;
* Регистрация заявок на добавление инсталляционных пакетов;
* Сохранений заявок в базе данных;
* Загрузка файлов на сервер;
* Просмотр каталога ПО;
* Просмотр информации о файле;
* Выгрузка файлов с сервера.

Для модератора:

* Сквозная аутентификация через Active Directory;
* Просмотр каталога ПО;
* Просмотр информации о файле;
* Загрузка файлов на сервер;
* Выгрузка файлов с сервера;
* Заполнение информации о файлах; тип информации будет уточнен в процессе разработки;
* Редактирование информации о файлах;
* Сохранение информации о файлах в базе данных;
* Просмотр списка файлов;
* Создание новых категорий для группировки файлов;
* Сохранение категорий в базе данных;
* Удаление категорий;
* Привязка файлов к категориям;
* Просмотр файлов, информация о которых еще не заполнена;
* Просмотр информации о пользователях; тип информации будет уточнен в процессе разработки.

Для администратора системы:

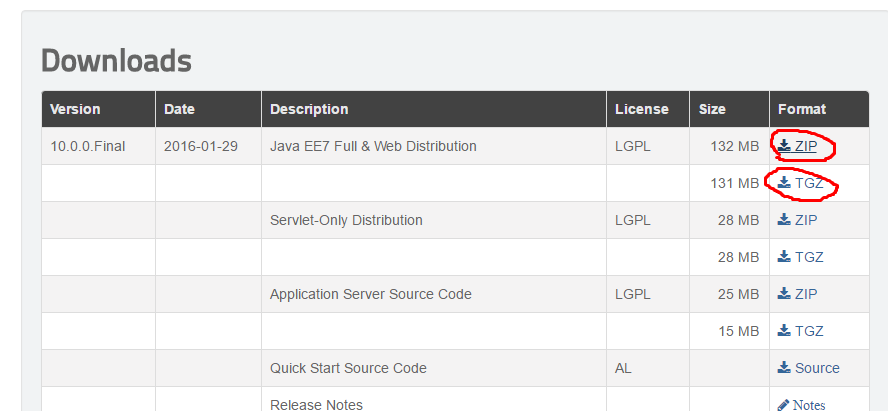
* Все возможности модератора;
* Изменить настройки;
* Удаление файлов;
* Выполнение скриптов для экспорта файлов в SCCM.

1. Java

Для начала разработки необходимо установить Java JDK, которую мы можем скачать [с официального сайта Oracle](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html).

1. WildFly

Для запуска приложения нужен сервер приложений WildFly. Скачать можно с [официального сайта](http://wildfly.org/downloads/).

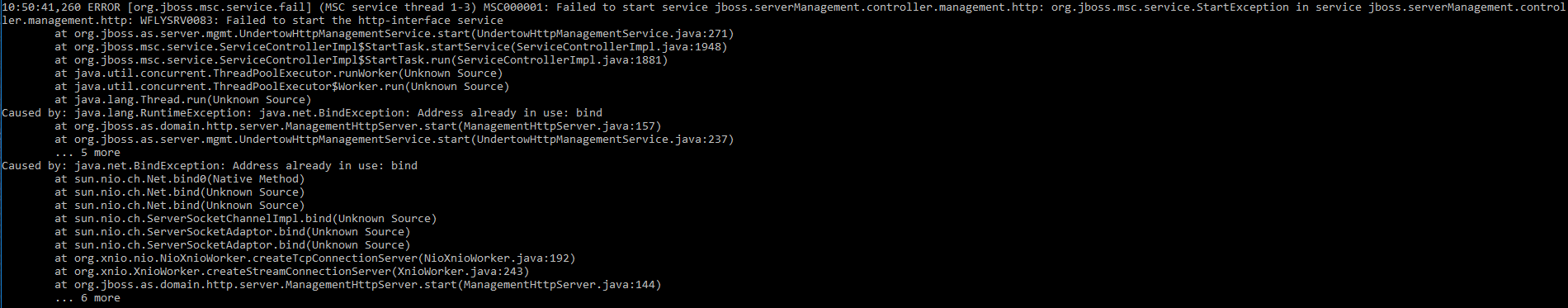


Скачиваем и распаковываем архив куда-нибудь.

Для запуска сервера необходимо запустить bin/standalone.bat.

* 1. Ошибка при запуске

Во время запуска на Windows может возникнуть ошибка вида:



Это значит, что требуемые для запуска порты заняты. Для того, чтобы изменить порты открываем standalone/configuration/standalone.xml, ищем внизу файла строки:

<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">

<socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"/>

<socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9993}"/>

<socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>

<socket-binding name="http" port="${jboss.http.port:8080}"/>

<socket-binding name="https" port="${jboss.https.port:8443}"/>

<socket-binding name="txn-recovery-environment" port="4712"/>

<socket-binding name="txn-status-manager" port="4713"/>

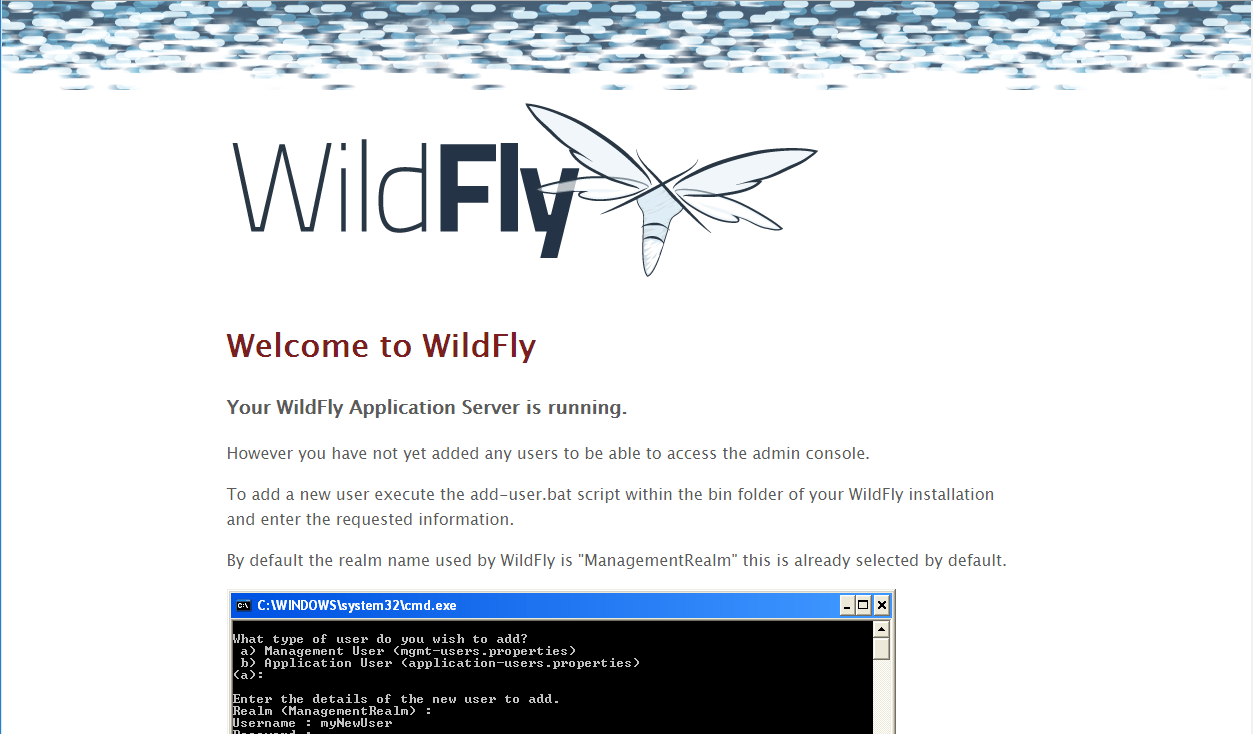
<outbound-socket-binding name="mail-smtp">

<remote-destination host="localhost" port="25"/>

</outbound-socket-binding>

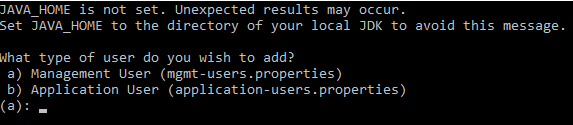
</socket-binding-group>

Необходимо изменить порты в ${jboss.management.http.port:9990} и ${jboss.management.https.port:9993} на другие. В данном случае, порт 9990 занят драйверами NVidia, поэтому меняем 9990 на 9991, сохраняем и запускаем сервер еще раз. Если все прошло без ошибок, то открываем <http://127.0.0.1:9991/>, и мы должны увидеть стартовую страницу:

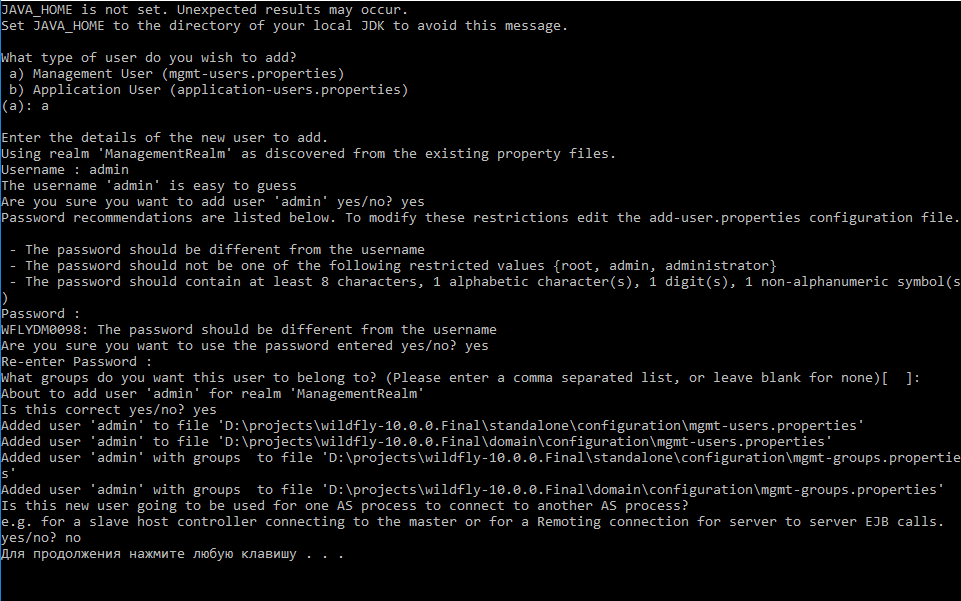


* 1. Добавление пользователя

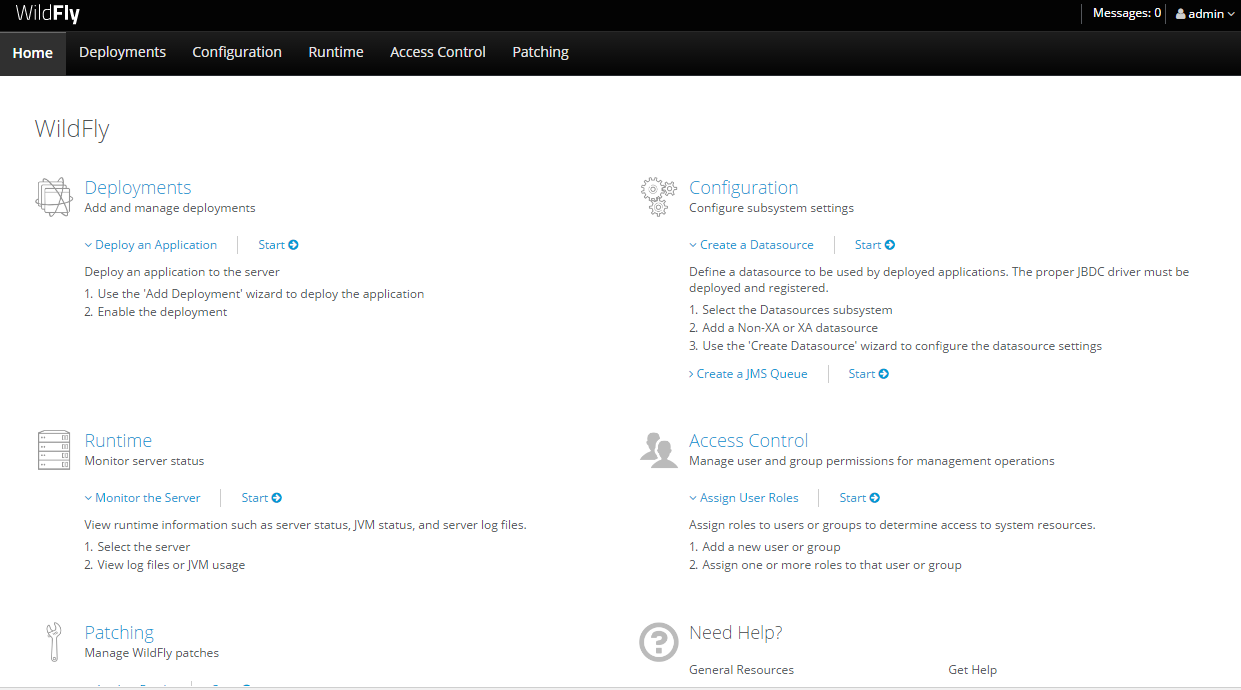
На стартовой странице написано, что нужно добавить пользователя. Для этого, запускаем /bin/add-user.bat.



Выбираем a) Management user и вводим все что у нас просят:



После добавления переходим по адресу <http://127.0.0.1:9991/>, где мы должны увидеть панель администратора:



1. Среда разработки

Будем рассматривать процесс разработки в среде IntelliJ Idea, которую мы можем скачать с [официального сайта](https://www.jetbrains.com/idea/download/).

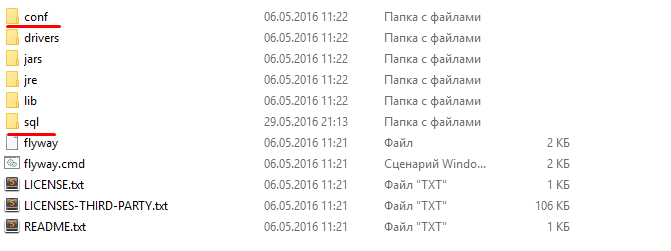
1. База данных

База данных создается с помощью миграции. Файл миграции лежит в /migrations/.

Для работы с миграциями необходимо скачать Flyway. У этого приложения есть инструмент для работы из командной строки. Скачать его можно на [официальном сайте](https://flywaydb.org/getstarted/download).



Рассмотрим на примере для Windows. Скачиваем и распаковываем архив. Нам нужны будут 2 папки:



Теперь нужно настроить адрес до БД. Заходим в распакованную папку и открываем файл conf/flyway.conf.

Находим строчку «flyway.url=». Выше, в комментариях есть примеры того, что надо писать для разных БД. Например, mariadb:

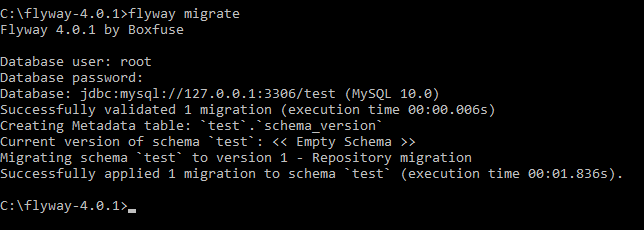
flyway.url=jdbc:mariadb://127.0.0.1:3306/test

Теперь копируем в папку sql/ файл миграции из проекта /migrations/V1\_\_Repository\_migration.sql.

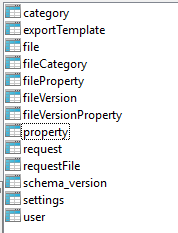
Запускаем cmd, переходим в распакованную папку, например, flyway-4.0.1 и выполняем команду:

flyway **migrate**

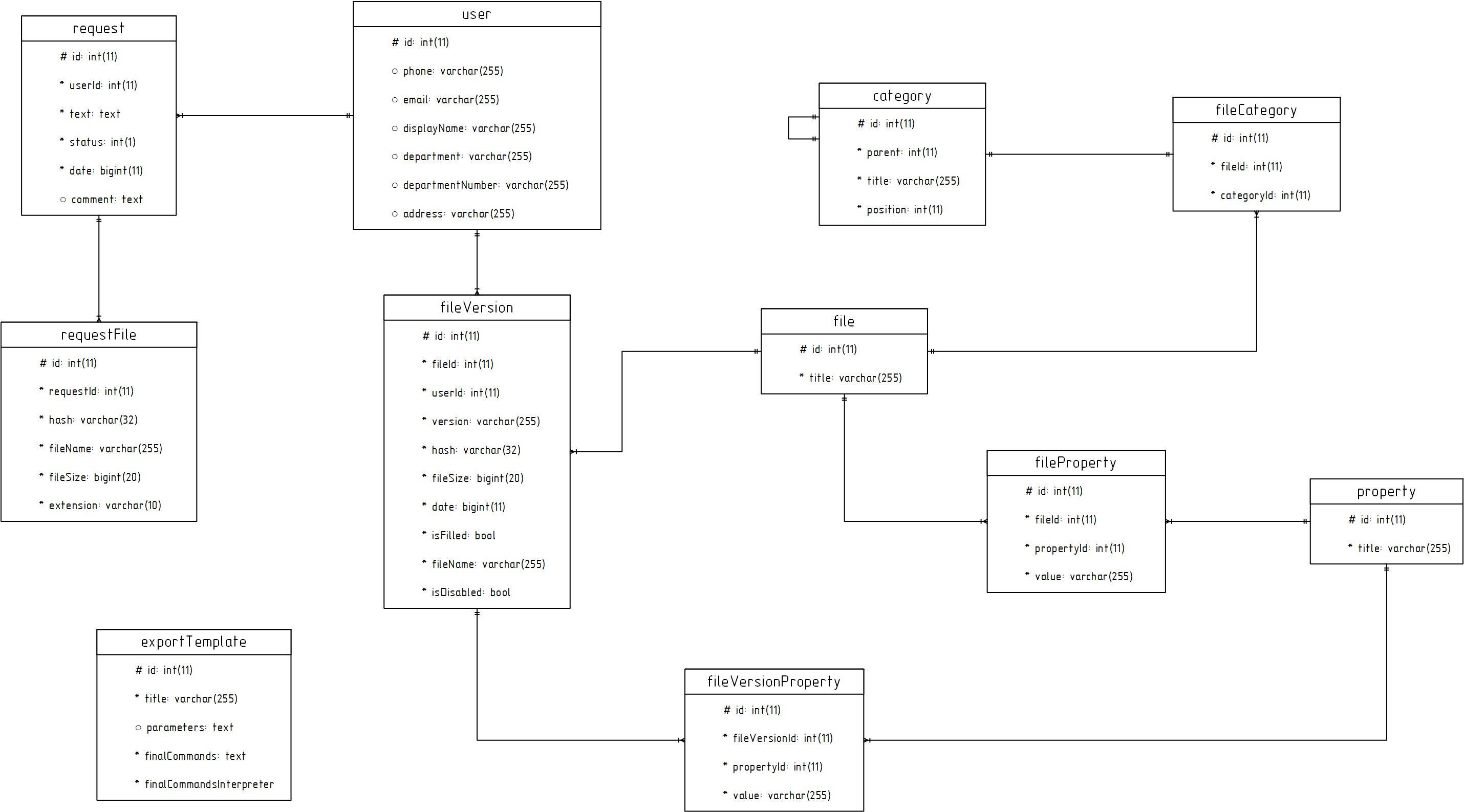
У нас попросят вести пользователя для БД и его пароль, после чего будет применена миграция:



Открываем БД и видим созданные таблицы:

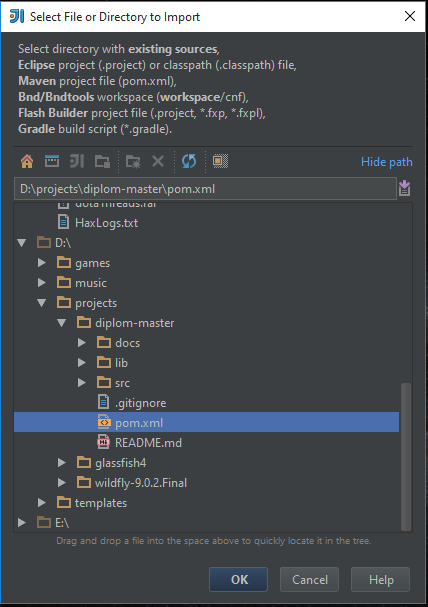


Логическая схема базы данных приведена ан рисунке:



1. Создание проекта в IntelliJ Idea

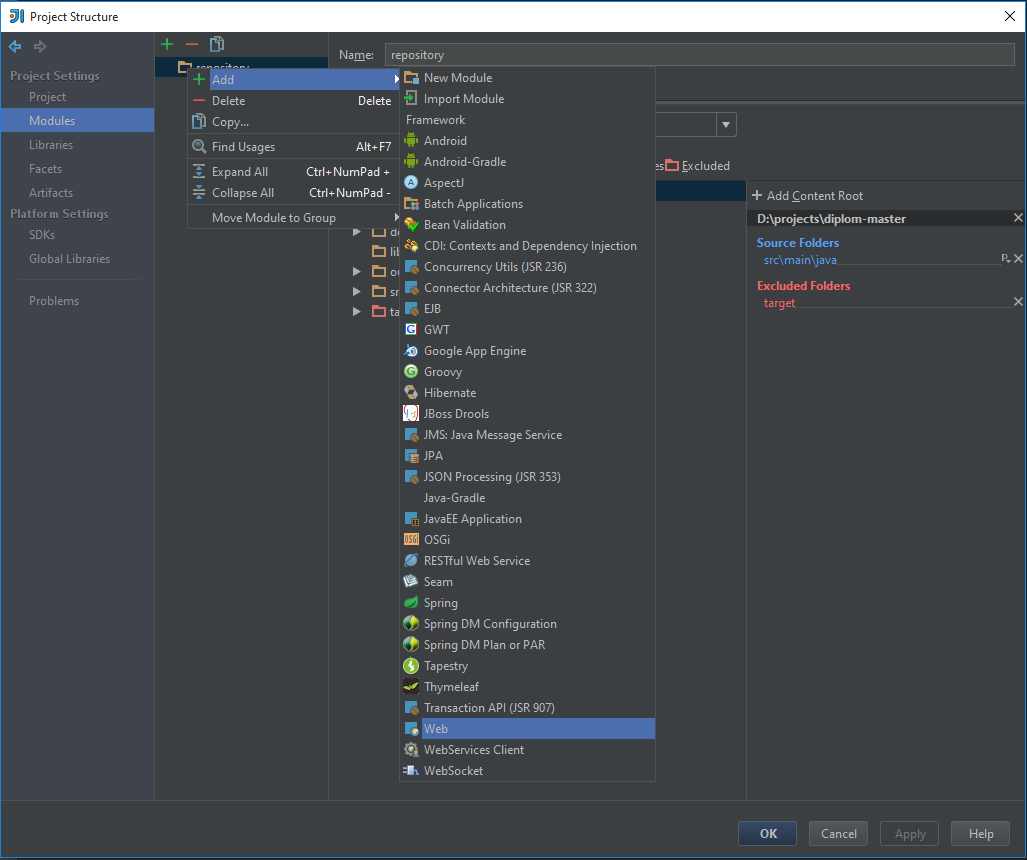
Для создания проекта нажимаем «Import project» и выбираем путь до исходных файлов, до файла pom.xml.

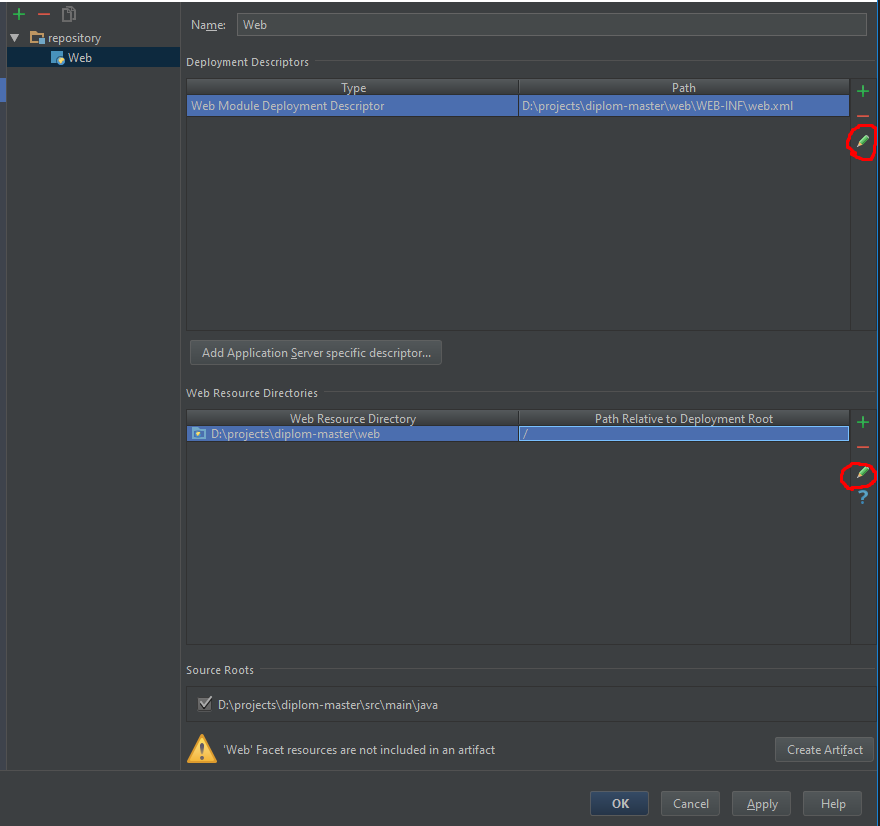


Жмем «ok», далее оставляем все без изменений, нажимаем «next», «next», «finish».

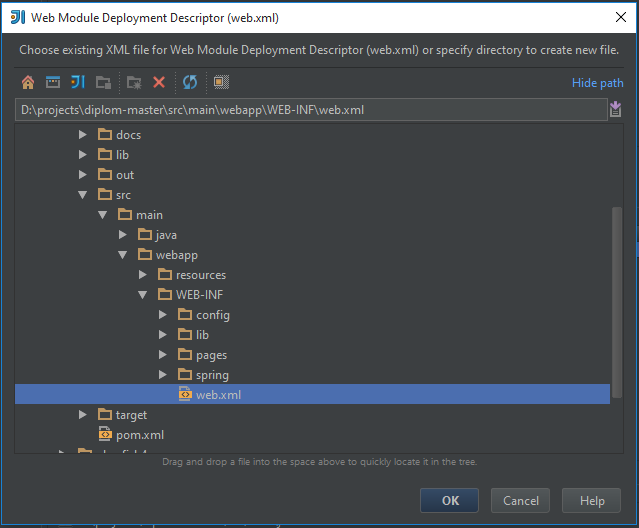
* 1. Создание модулей

Нажимаем «File» -> «Project Structure» -> «Modules», правой кнопкой по «repository» -> «add» -> «Web».

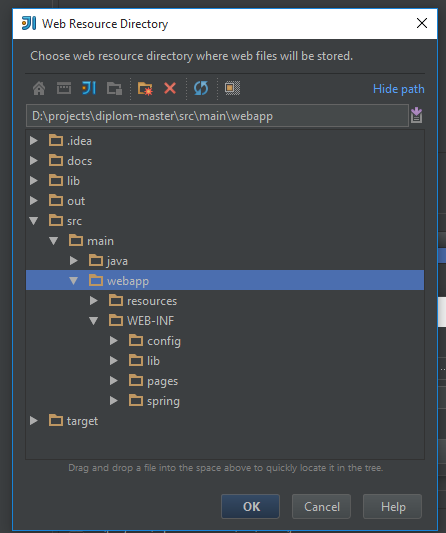


Далее необходимо настроить пути до папки webapp. 

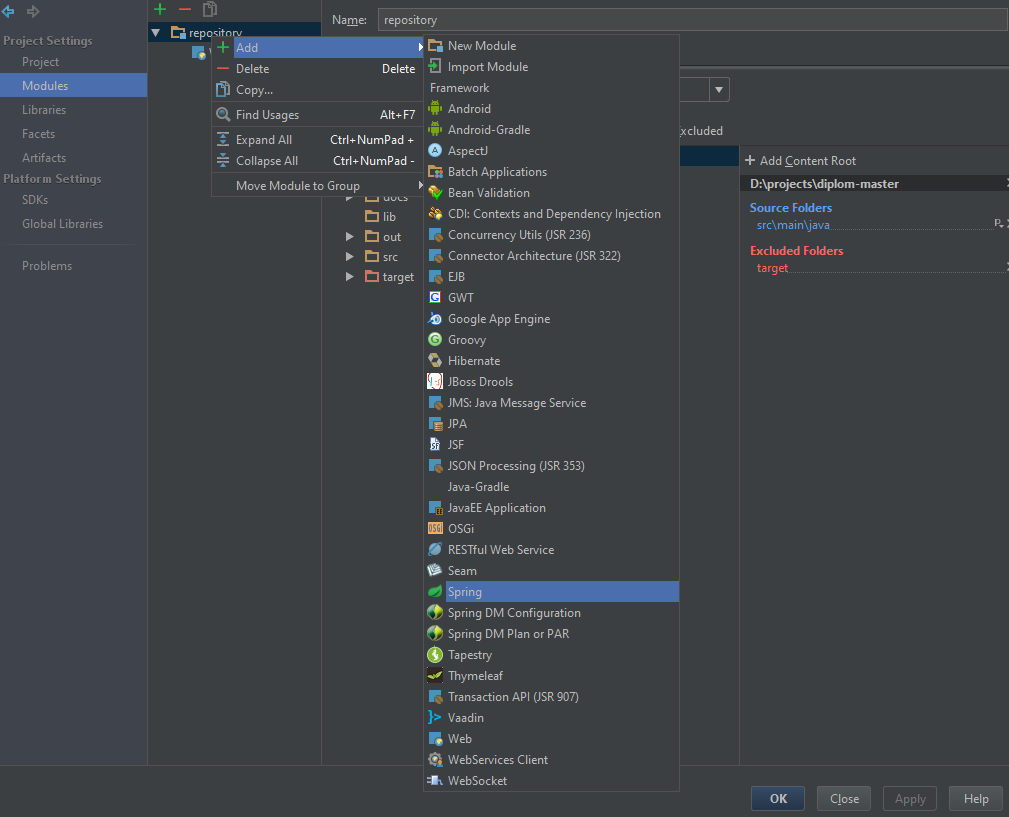
В верхнем пути выбираем путь до web.xml:



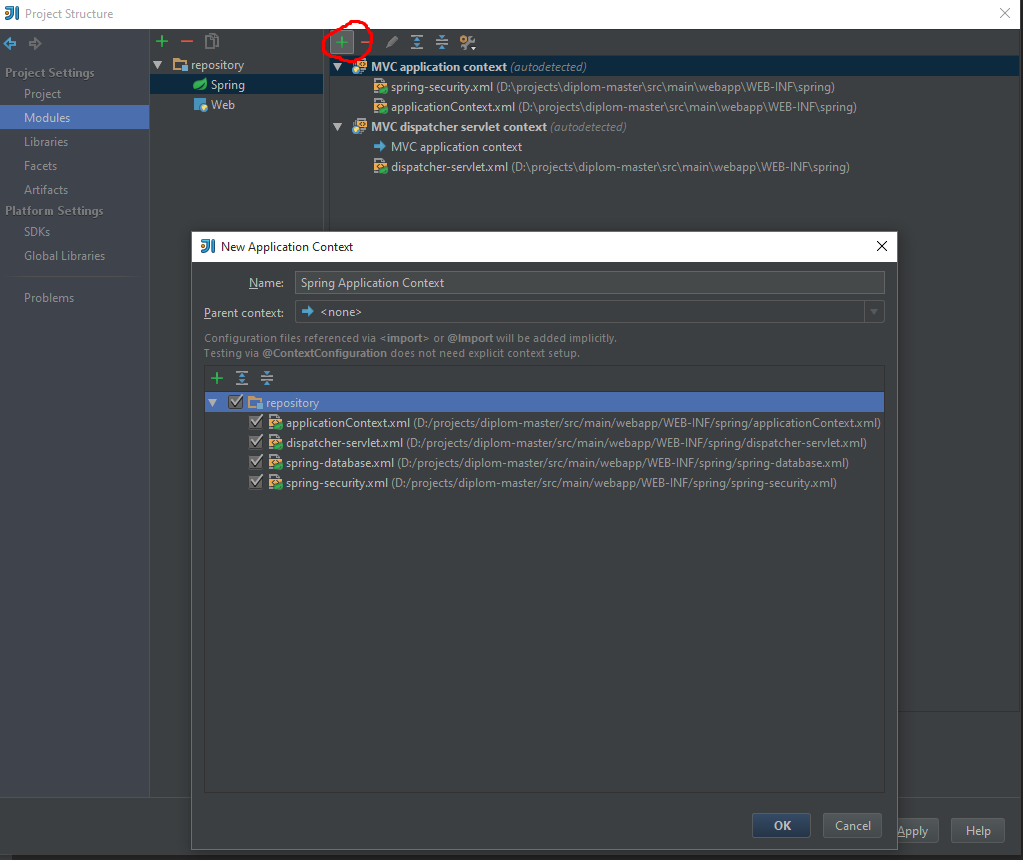
В нижнем – путь до папки webapp:



Добавляем еще 1 модуль Spring

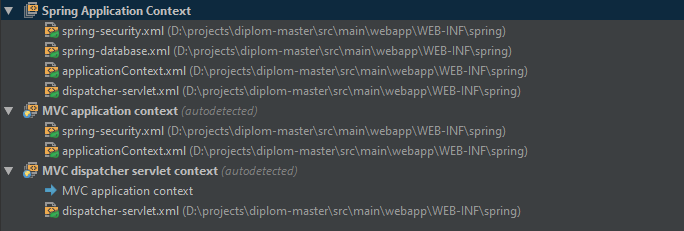


Нажимаем на «+»



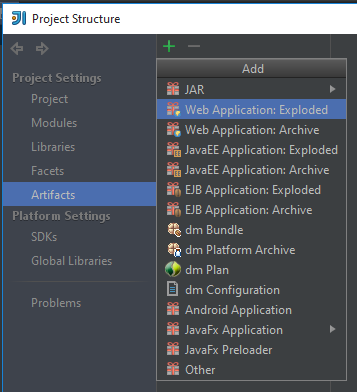
В появившемся окне выбираем все файлы и нажимаем «ok». Далее еще раз «ok».

В результате у модуля Spring должна быть следующая конфигурация

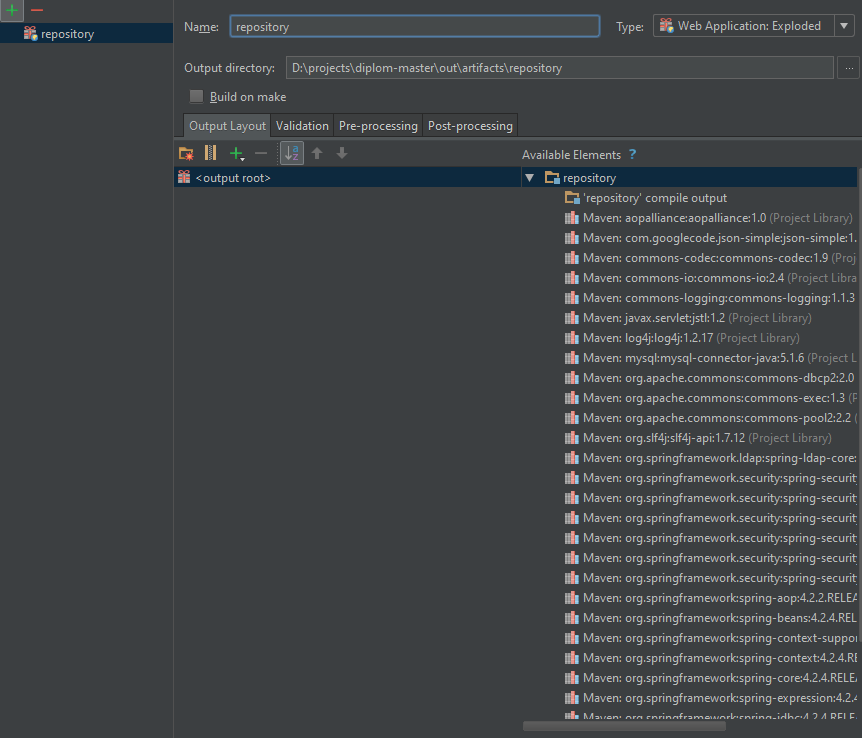


* 1. Создание артефакта

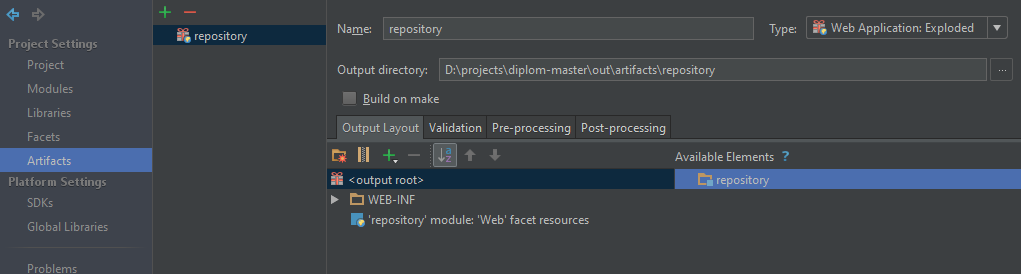
Необходимо создать артефакт, который будет деплоится на сервер приложений. Для этого, нажимаем «File» -> «Project Structure» -> «Artifacts» -> «+» -> «Web Application: Exploded».



Вводим любое название в поле Name



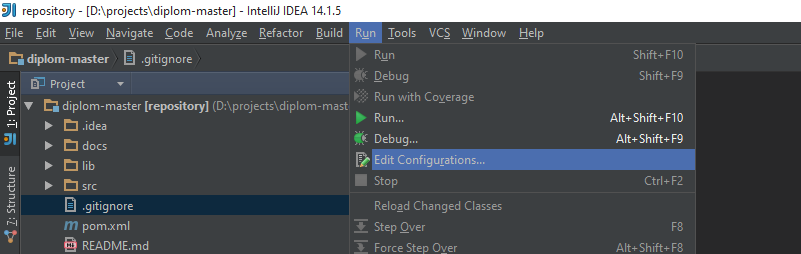
Нажимаем на «repository» справа правой кнопкой и, затем, «Put into output root», после чего библиотеки справа должны переместится в артефакт:



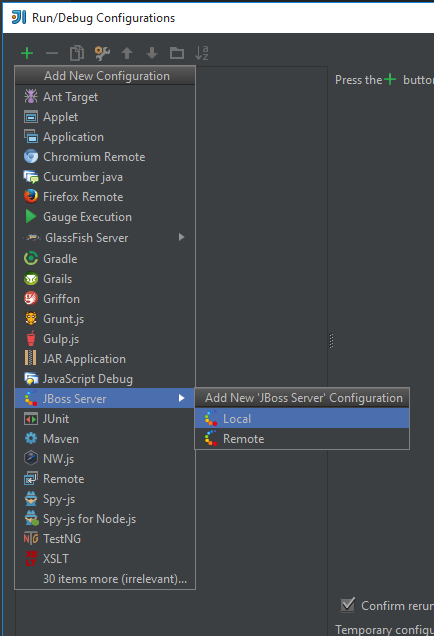
Артефакт создан, нажимаем «ok».

* 1. Создание конфигурации запуска

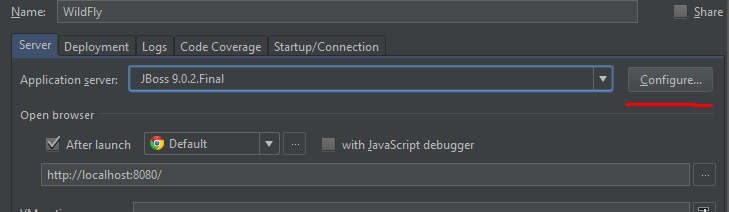
Далее, для запуска приложения, необходимо создать конфигурацию для запуска и добавить в Idea скачанный ранее сервер приложений WildFly. Для этого, нажимаем «Run» -> «Edit configuration».



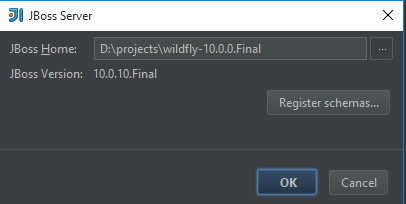
Далее, «+» -> «JBoss Server» -> «local»



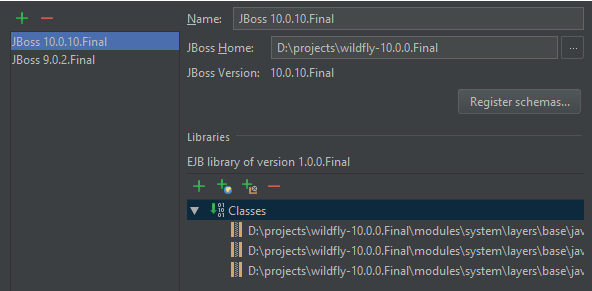
Нажимаем «Configure»



В появившемся окне нажимаем «+», выбираем путь до папки с сервером, в который распаковывали его в пункте 3 и нажимаем «ok».

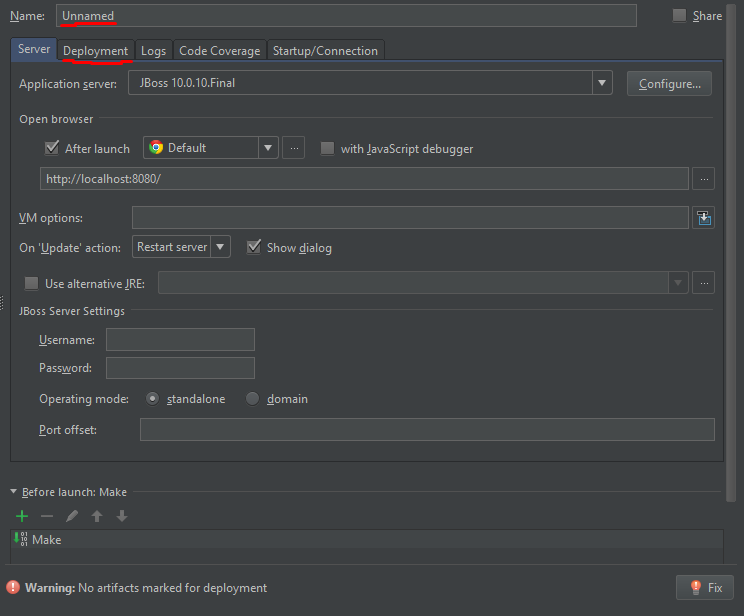


В списке сервером должен появится этот сервер. Выбираем его и нажимаем «ok».

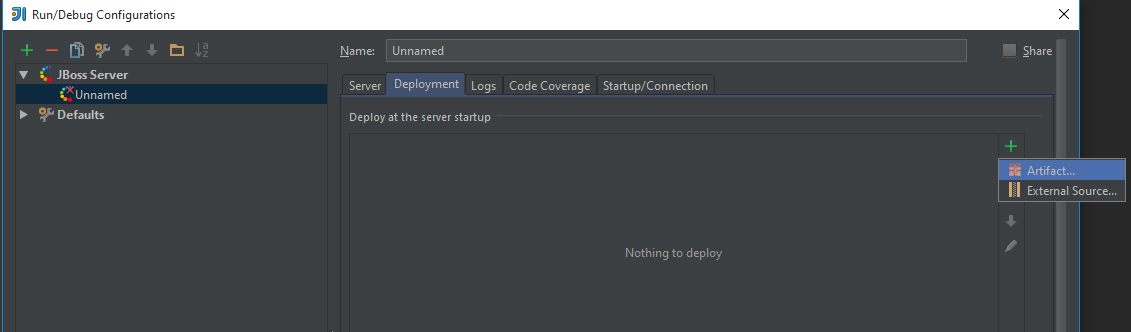


Далее продолжаем заполнять настройки конфигурации для запуска

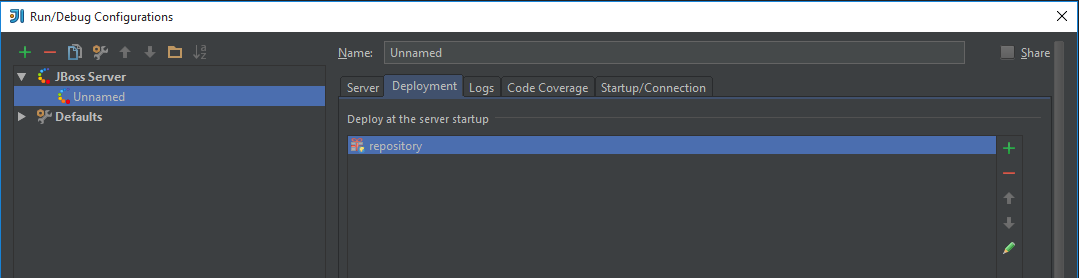
В поле Name можно ввести любое название. Далее переходим во вкладку «Deployment»:



Нажимаем на «+» справа -> «Artifact…»



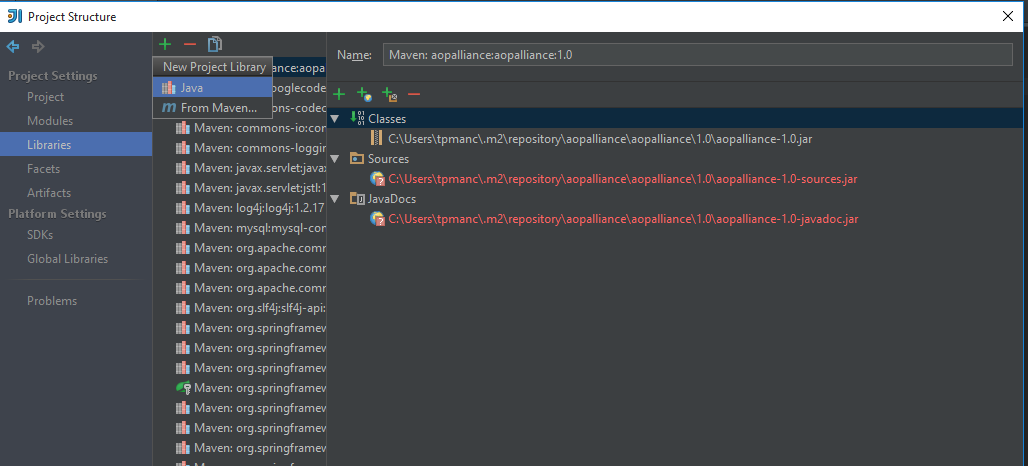
После этого должен появится созданный нами в пункте 4 артефакт.



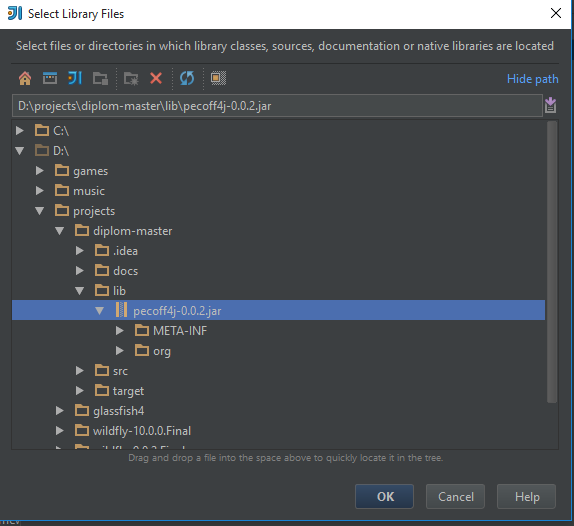
Нажимаем «ok», после чего конфигурация будет создана.

* 1. Добавление библиотеки

Необходимо добавить 1 библиотеку вручную. Нажимаем «File» -> «Project Structure» -> «Libraries» -> «+» -> «Java».

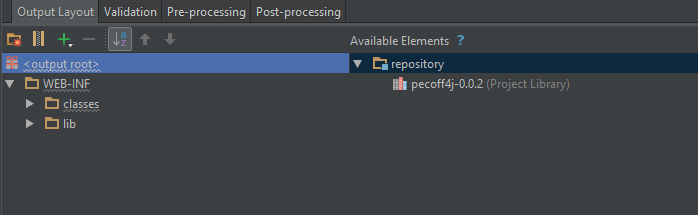


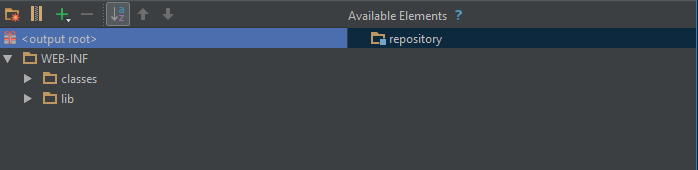
В появившемся окне выбираем путь до папки с проектом, файл lib/pecoff4j-0.0.2.jar и нажимаем «ok».



После добавления библиотеки ее необходимо добавить в артефакт. Также, любую другую библиотеку, которую мы добавили в проект, необходимо добавлять в артефакт.

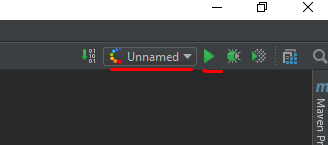
«File» -> «Project Structure» -> «Artifacts» и двойным кликом по библиотеке перемещаем ее влево.



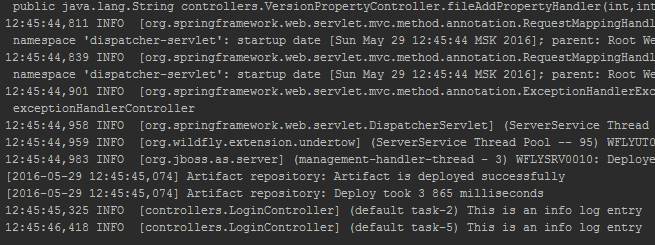


* 1. Запуск

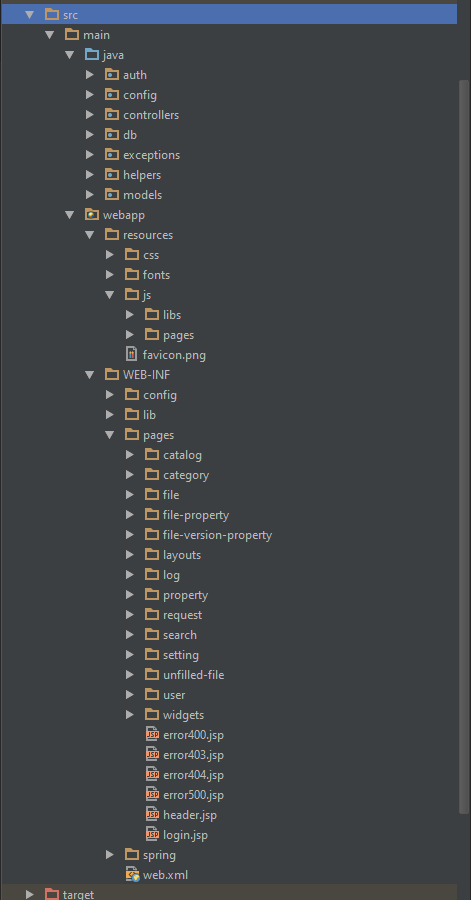
Теперь можно запустить приложение. Справа сверху выбираем название нужной конфигурации и жмем пуск.



Idea сама запустит сервер приложений а задеплоит артефакт на него. При успешном запуске в консоли выведется следующий текст:



1. Структура проекта

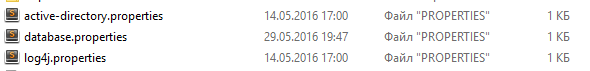


Весь код лежит в папке src/main/java.

* auth – классы для работы с Active Directory
* controllers - контроллеры
* models - модели
* helpers – вспомогательные классы

Виды лежат в src/webapp/WEB-INF/pages/название модели/.

При первом запуске в домашней папке пользователя создается папка repository, в которой создаются файлы конфигурации БД, Active Directory и логирования.



* 1. Контроллеры

Контроллеры должны помечаться аннотацией @Controller.

Для того, чтобы привязать метод контроллера к какому-либо URL, нужно добавить аннотацию, например,

@RequestMapping(value = {"/catalog", "/"}, method =RequestMethod.GET)

Каждый метод контроллера должен возвращать строку, в которой будет путь до нужного файла jsp вида относительно папки src/main/webapp/WEB-INF/pages. Например, метод возвращает строку "catalog/catalog". Тогда будет использоваться файл src/main/webapp/WEB-INF/pages catalog/catalog.jsp.

Для того, чтобы передавать данные в вид, нужно в список входных параметров метода контроллера добавить Model model.

public String users(  
 @RequestParam(value="categoryId", required=false, defaultValue = "0") int categoryId,  
 @RequestParam(value="page", required=false, defaultValue = "1") int page,  
 Principal principal,  
 Model model  
)

Теперь, для передачи данных в шаблон пишем

model.addAttribute("requestCount", requestCount);

Использовать это в шаблоне можно так:

<h2>**${**requestCount**}**</h2>

* 1. Модели

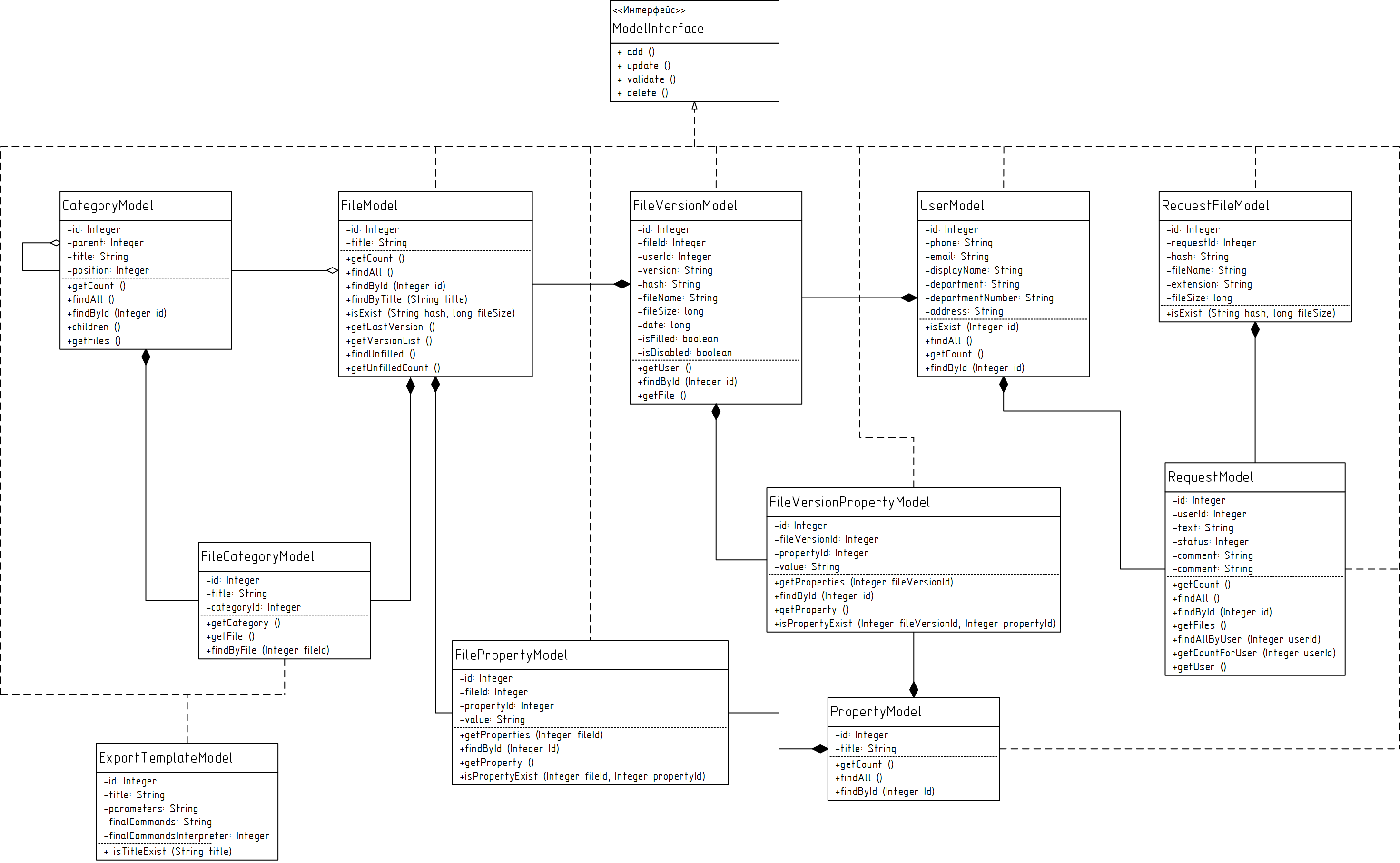
Все модели реализуют интерфейс ModelInterface. Необходимо реализовать методы: add(), validate(), delete(), update().

Для работы в БД пишем:

NamedParameterJdbcTemplate template = new NamedParameterJdbcTemplate(Database2.*getInstance*().getBds());  
MapSqlParameterSource parameters = new MapSqlParameterSource();  
parameters.addValue("id", id);  
List<Map<String, Object>> rows = template.queryForList(*getById*, parameters);  
for (Map row : rows) {  
 Integer fileId = (Integer) row.get("id");  
 String title = (String) row.get("title");  
 return new FileModel(fileId, title);  
}

Здесь создается шаблон и набор параметров, которые мы хотим подставить в SQL запрос.

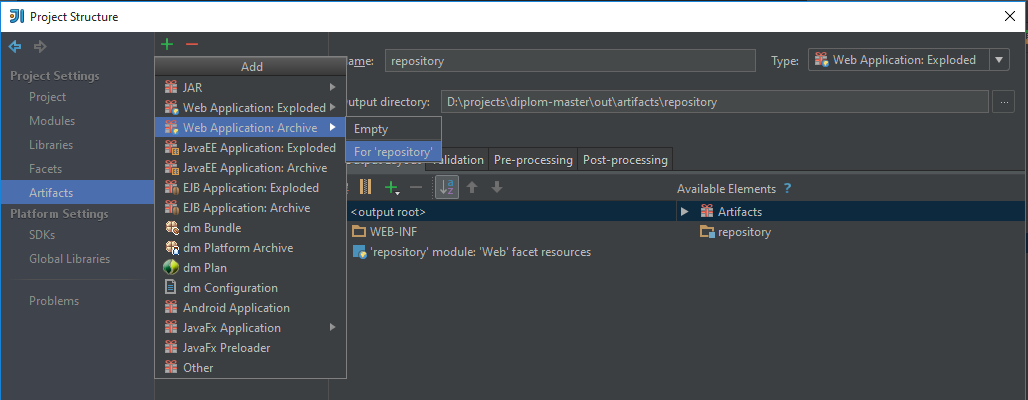
Диаграмма классов:



1. Билд

В результате билда нам нужно получить war архив. Для этого, надо создать еще один артефакт:

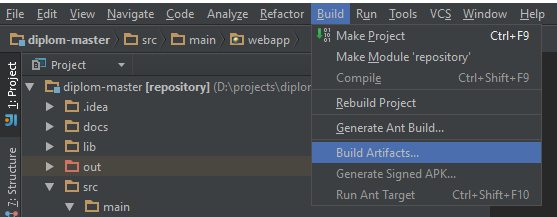
«File» -> «Project Structure» -> «Artifacts» -> «+» -> «Web Application: Archive»



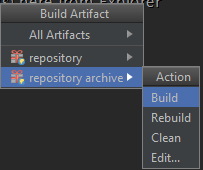
Обязательно выбираем For «repository». Вместо repository может быть другое название, если предыдущий артефакт был назван по-другому. Нажимаем «For «repository»», создается артефакт и нажимаем «ok», чтобы сохранить его.

Теперь сам билд:

Кнопка сверху «Build» -> «Build Artifacts»



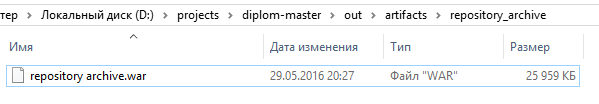
В появившемся окошке выбираем только что созданный артефакт и Build.



Результат билда написан в самом низу:



Идем в папку с проектом, out/artifacts/repository\_archive и видим нужный нам war файл:

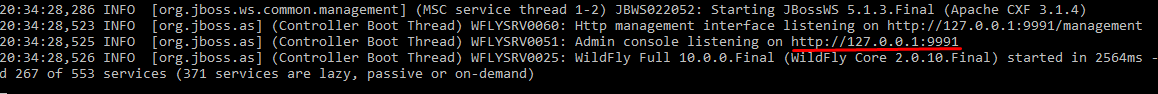


Его можно деплоить на сервер приложений.

1. Работа с WildFly
   1. Деплой на сервер приложений

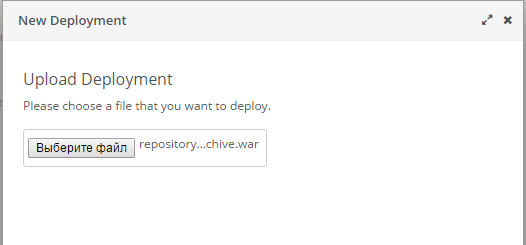
Должен быть запущен сервер приложений. Пример запуска написан в пункте 3. Если коротко, то нужно запустить bin/standalone.bat.

Заходим в панель администратора. Ее адрес пишется в консоль, в самом конце, при запуске сервера.

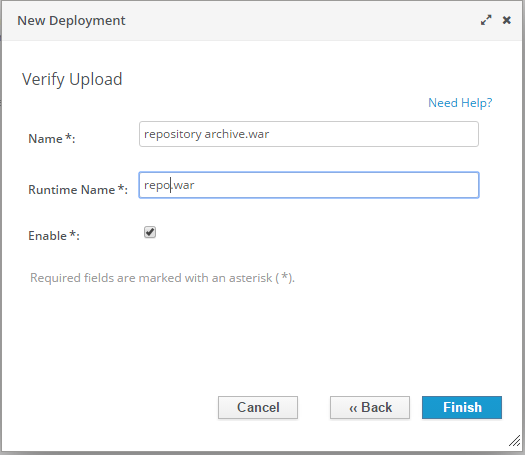


Заходим во вкладку Deployments и видим слева список задеплоенных приложений, которые появились из-за того, что их добавила IntelliJ Idea. Для добавления нового нажимаем «add», выбираем «Upload a new deployment».

Выбираем полученный ранее war файл:



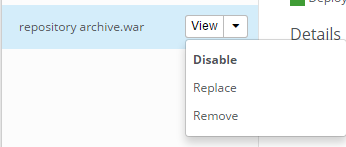
На следующей форме вводим в поле Name уникальное имя, а в поле Runtime Name адрес, по которому будет открываться приложение, например:



Нажимаем Finish. Теперь приложение доступно по адресу <http://localhost:8080/repo/>.

* 1. Перезагрузка приложения

Перезагрузка может понадобится, например, когда необходимо поменять настройки в приложении. Для перезагрузки заходим в админку WildFly, переходим на вкладку Deployments. Выбираем нужное приложение и жмем Disable:



Теперь включаем:

