**1 Начало работы с Java Spring**

**1.1 Spring Framework**

Spring — это легкая среда разработки приложений с открытым исходным кодом, использующая Java Enterprise Edition (Java EE) и модель Enterprise JavaBeans (EJB). Он поддерживает другие интегрированные фреймворки в экосистеме Java, такие как Struts, Hibernate и Tapestry. Spring поставляется во многих вариантах, которые адаптированы для конкретных целей, таких как Spring MVC, Spring Boot или Spring Security.

Spring известен своими системами внедрения зависимостей и инверсии управления (IoC), которые позволяют с легкостью создавать крупномасштабные слабосвязанные приложения. Spring особенно подходит для финансовых и корпоративных приложений благодаря своей скорости, безопасности и простоте построения транзакционных систем.

1.1.1 Преимущества Spring

1. Объект Java (POJO - Plain Old Java Object): приложения Spring просто используют обычные объекты Java, а не серверы или корпоративные контейнеры. Это делает приложения более легкими.
2. Модульность IoC и MVC позволяют повторно использовать компоненты в вашем приложении без ручного управления зависимостями каждого из них.
3. Сильная поддержка экосистемы Java: Spring новаторски использует уже существующие технологии, такие как ORM-фреймворки, JEE и таймеры JDK.
4. Масштабируемые транзакции: Spring предлагает последовательный, масштабируемый интерфейс управления транзакциями как для локальных, так и для глобальных транзакций.
5. Включает протоколы кеширования.
6. Безопасность: Простота внедрения модулей безопасности с функциями аутентификации и проверки.
7. Гибкие конфигурации: можно использовать аннотации на основе Java или конфигурацию XML.

1.1.2 Недостатки Spring

1. Сложность: Spring Framework предпочитает повышенный контроль и сложность простоте использования. Поэтому Spring сложнее подобрать, чем другие веб-фреймворки. Несмотря на то, что Spring очень мощный, он лучше всего подходит для опытных разработчиков, которые могут справиться с его крутой кривой обучения.
2. Параллельные механизмы: широкий спектр возможностей Spring означает, что одну задачу можно выполнить несколькими способами. Потребуются знания доступных инструментов, чтобы выбрать идеальное решение и избежать путаницы между командами.
3. Отсутствие надежной документации: в документации Spring отсутствуют четкие рекомендации по ряду тем, особенно по методам кибербезопасности. Пользователи должны бороться с распространенными атаками, такими как межсайтовый подлог и атаки с использованием сценариев.

Многие из этих недостатков были устранены с помощью изобретения модуля Spring Boot, который предназначен для микросервисных архитектур.

1.1.3 Особенности Spring

1. Инверсия управления (IoC): убирает зависимости с объектов на сам фреймворк. Без IoC каждый объект должен обрабатывать свой собственный экземпляр зависимости. В IoC объект просто указывает, что ему нужна определенная зависимость, и платформа настраивает и управляет ею автоматически. Это упрощает процесс разработки приложения за счет консолидации системы зависимостей и снижения нагрузки на отдельные классы.
2. Аспектно-ориентированное программирование: Spring AOP позволяет иметь разные загрузчики классов для повышения модульности и разделения сквозных проблем.
3. Внедрение зависимостей: процесс, который Spring использует для автоматического управления зависимостями bean-компонентов. Spring проверяет, какие bean-компоненты необходимы для работы конкретного bean-компонента, и внедряет их в качестве зависимости. Spring может выполнять внедрение зависимостей с помощью конструктора или метода установки.
4. Управление жизненным циклом контейнера: автоматическое создание, управление и настройка объектов приложения.
5. Поддержка MVC: поддерживает веб-архитектуру MVC, которая разделяет функциональные возможности между уровнями модели, представления и контроллера.
6. Обработка ошибок: включает обработку исключений JDBC с системой иерархии исключений.

**1.2 Java Spring: начало работы Java Backend API**

1.2.1 Подготовка к созданию проекта

Первое, что нужно сделать, это убедиться, что загружены и установлены все необходимые инструменты. Для разработки требуется установить Java JDK. Oracle имеет хорошую документацию для его установки на компьютер.

Среду разработки можно использовать Visual Studio Code или Eclipse, в примере рассматривается работа в VS.

Для удобства разработки в VS можно установить следующие расширения:

* Spring Boot Dashboard;
* Spring Boot Tools;
* Spring Initializr Java Support;
* Java IDE;
* Java Extension Pack;
* Java Dependency Viewer;
* Debugger for Java.

В VS Code создается проект Spring Maven на языке Java (рис. 1.1).

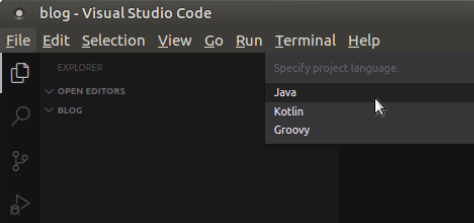


Рис. 1.1 – Выбор языка

Далее в приглашении будет запрошен идентификатор группы (Group ID). Для приложения такого размера можно указать «javapractice» (рис. 1.2).

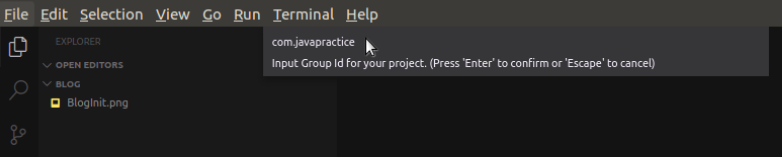


Рис. 1.2 – Выбор идентификатора группы

Он нужен для связывания различных частей проекта между собой.

Затем будет предложено ввести идентификатор артефакта (Artifact Id). Это специальное название части приложения, для проекта можно оставить «demo» (рис. 1.3).

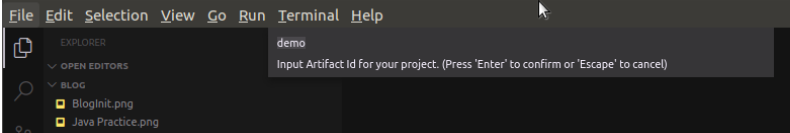


Рис. 1.3 - Выбор идентификатора артефакта

Затем нужно указать версию Spring, значение по умолчанию 2.2.6.

Также можно сгенерировать файл пакета с помощью <https://start.spring.io>. Он создает основу для приложения и позволяет загружать ее в виде zip-файла, который открывается в редакторе кода.

1.2.2 Зависимости

Зависимости (Dependencies) - это наборы методов и классов, которые будут использоваться в проекте, после импорта. Например, две основные зависимости: Spring Web и Spring Boot DevTools (рис. 1.3).

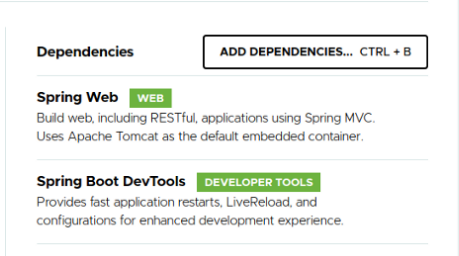


Рис. 1.3 – Зависимости Spring Web и Spring Boot DevTools

Для Spring Initializr в VS Code (рис. 1.4).

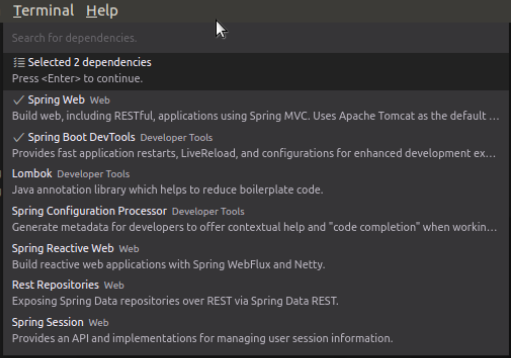


Рис. 1.4 – Зависимости в VS

1.2.3 Структура

После создания проекта в VS будет сгенерирована структура в указанной папке (рис. 1.5).

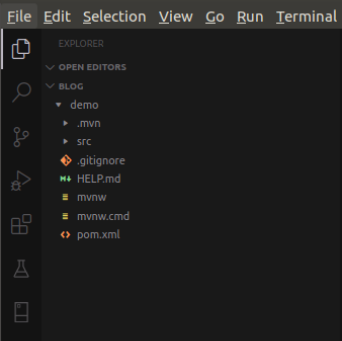


Рис. 1.5 – Структура проекта

Java Spring - это очень легкий фреймворк, который исключает ненужный код, который не используется. Spring Framework добавлен, этого достаточно для запуска проекта. Тем самым сокращается общий размер приложения.

Большая часть кода будет содержаться в папке src, в которой содержатся папки main и test.

В папке main хранится весь код, который используется для создания проекта. В папке test хранятся любые тестовые файлы, которые создаются во время разработки на основе тестирования (рис. 1.6).

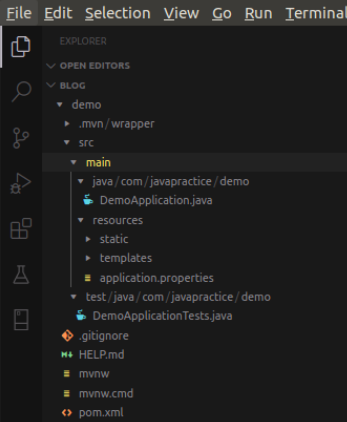


Рис. 1.6 – Содержимое папки src

В папке main присутствует файл DemoApplication.java. Это основной файл для приложения Spring. Этот файл свяжет все файлы в проекте друг с другом, чтобы создать работающее приложение.

В папку проекта com.javapractice.demo нужно добавить следующие папки:

* entities - в этой папке будут содержаться модели;
* dao — эта папка будет содержать служебные файлы, которые позволяют приложению взаимодействовать с базой данных;
* config — эта папка будет содержать файлы конфигурации;
* controller — эта папка будет содержать файлы контроллера, которые позволят пользователю взаимодействовать с приложением.

Макет приложения выглядит как на рис. 1.7.

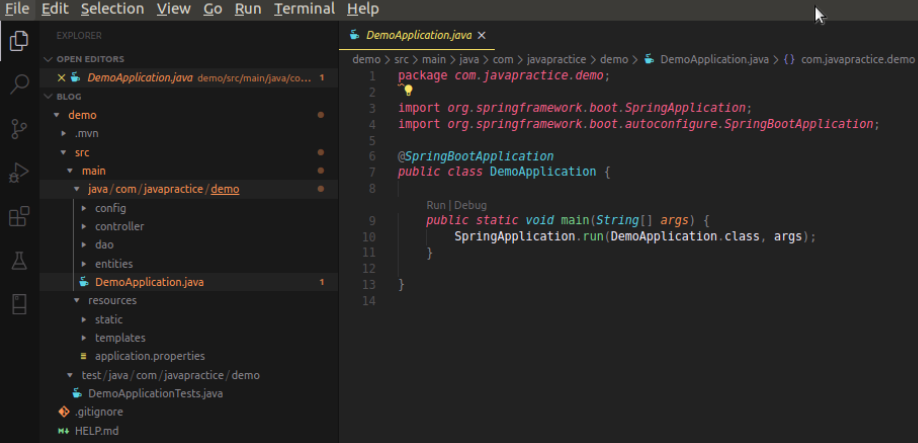


Рис. 1.7 - Макет приложения

В файле pom.xml в корневой папке проекта, прописываются все зависимости. Файл выглядит следующим образом:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" |
|  | xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"> |
|  | <modelVersion>4.0.0</modelVersion> |
|  | <parent> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-parent</artifactId> |
|  | <version>2.2.6.RELEASE</version> |
|  | <relativePath/> <!-- lookup parent from repository --> |
|  | </parent> |
|  | <groupId>com.javapractice</groupId> |
|  | <artifactId>demo</artifactId> |
|  | <version>0.0.1-SNAPSHOT</version> |
|  | <name>demo</name> |
|  | <description>Demo project for Spring Boot</description> |
|  | <properties> |
|  | <java.version>1.8</java.version> |
|  | </properties> |
|  | <dependencies> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-web</artifactId> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-devtools</artifactId> |
|  | <scope>runtime</scope> |
|  | <optional>true</optional> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-test</artifactId> |
|  | <scope>test</scope> |
|  | <exclusions> |
|  | <exclusion> |
|  | <groupId>org.junit.vintage</groupId> |
|  | <artifactId>junit-vintage-engine</artifactId> |
|  | </exclusion> |
|  | </exclusions> |
|  | </dependency> |
|  | </dependencies> |
|  | <build> |
|  | <plugins> |
|  | <plugin> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-maven-plugin</artifactId> |
|  | </plugin> |
|  | </plugins> |
|  | </build> |
|  | </project> |

Далее требуется добавить все зависимости, которые понадобятся для проекта. Их добавлять в проект "на лету", добавляя их в раздел dependencies файла pom.xml. Необходимы зависимости:

|  |
| --- |
| <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-data-jpa</artifactId> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.flywaydb</groupId> |
|  | <artifactId>flyway-core</artifactId> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.postgresql</groupId> |
|  | <artifactId>postgresql</artifactId> |
|  | <scope>runtime</scope> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.projectlombok</groupId> |
|  | <artifactId>lombok</artifactId> |
|  | <optional>true</optional> |
|  | </dependency> |

Эти зависимости упростят подключение приложения к базе данных, а также сократят часть кода.

После сохранения файла, он автоматически обновит приложение.

Тестовый запуск приложения, через терминал, командой:

./mvnw spring-boot:run

Приложение выдаст ошибку, потому что не настроена и не подключена база данных (БД) к приложению. Однако при этом будут загружены и установлены все необходимые файлы, необходимые приложению для запуска.

**1.3 Установка PostgreSQL**

Реляционные системы управления базами данных (РСУБД) - это ключевой компонент многих веб-сайтов и приложений. Они обеспечивают структурированный способ хранения данных и организацию доступа к информации. PostgreSQL- это объектно-реляционная система управления базами данных, которая все больше и больше вытесняет MySQL и производственных серверов.

Её преимущество в множестве дополнительных функций и улучшений, таких как надежная передача данных и параллелизация без блокировок чтения. Можете использовать эту СУБД с различными языками программирования, а её синтаксис запросов PL/pgSQL очень похож на MySQL от Oracle.

1.3.1 Установка из официальных репозиториев

Это очень популярная СУБД, потому программа присутствует в официальных репозиториях (в примере рассмотрена установка на Linux Ubuntu 20.04). Для установки используются следующие команды. Команды необходимо выполнять от имени администратора с помощью sudo. Сначала требуется обновить списки пакетов:

sudo apt update

Установка СУБД PostgreSQL (рис. 1.8):

sudo apt -y install postgresql

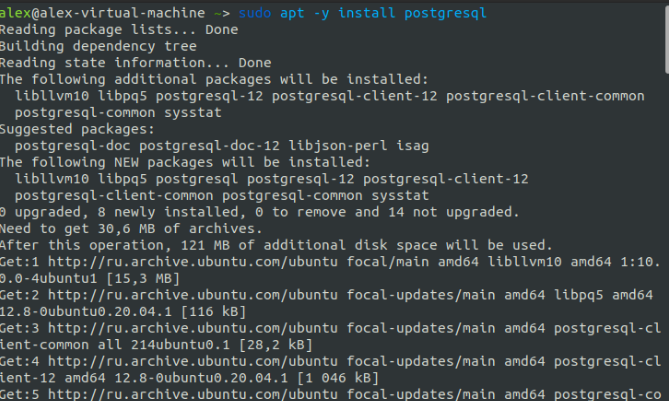


Рис. 1.8 - Установка СУБД PostgreSQL

1.3.2 Настройка PostgreSQL

После установки СУБД в терминале надо переключится на пользователя postgres с помощью команды:

sudo -i -u postgres

Эта учетная запись создается во время установки программы и на данный момент можно получить доступ к системе баз данных только с помощью нее. По умолчанию PostgreSQL использует концепцию ролей для аутентификации и авторизации.

Это очень похоже на учетные записи Unix, но программа не различает пользователей и групп, есть только роли. Сразу после установки PostgreSQL пытается связать свои роли с системными учетными записями, если для имени системной учетной записи существует роль, то пользователь может войти в консоль управления и выполнять позволенные ему действия. Таким образом, после переключения на пользователя postgres можно войти в консоль управления (рис. 1.9):

psql

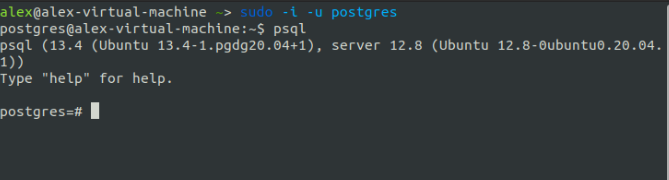


Рис. 1.9 – Ввод команд от пользователя postgres

Просмотр информации о соединении:

\conninfo

Чтобы выйти:

\q

1.3.3 Создание роли PostgreSQL

Учетная запись postgres является администратором, поэтому имеет доступ к функциям управления. Команда для создания пользователя:

createuser --interactive

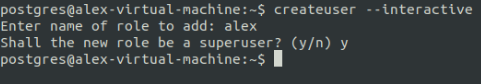


Рис. 1.10 – Создание пользователя

Скрипт задаст лишь два вопроса, имя новой роли и нужно ли делать ее суперпользователем.

1.3.4 Создание базы данных

Точно также как имена ролей сопоставляются с системными пользователями, имя базы данных будет подбираться по имени пользователя. Например, если создать пользователя alex, то по умолчанию система попытается получить доступ к базе данных alex. Она создается командной:

createdb alex

Дальше, чтобы подключиться к этой базе данных нужно войти от имени одноименного пользователя:

sudo su - alex

Проверка информации о подключении в консоли (рис. 1.11):

psql

\conninfo

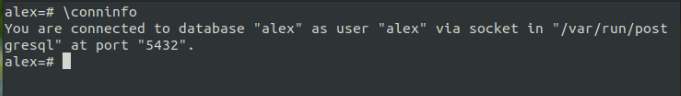


Рис. 1.11 - Проверка информации о подключении

Подключение прошло успешно с помощью роли alex к базе alex. Если нужно указать другую базу данных, можно сделать это с помощью опции - d, например:

psql -d postgres

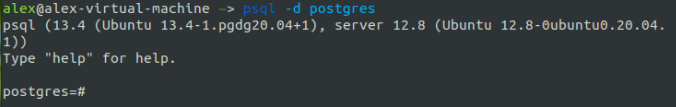


Рис. 1.12 – Подключение к другой БД

**1.4 Подключение БД к проекту**

Spring подключается к базе данных как к серверу, поэтому необходимо настроить Postgresй. Порт по умолчанию, который использует postgres, является <http://localhost:5432>.

После того, как настроен сервер базы данных, можно перейти к настройке приложения для подключения к нему.

В папке main/resources есть файл с именем "application.properties", содержащий следующий код:

spring.datasource.url=

spring.datasource.username=

spring.datasource.password=

spring.jpa.properties.hibernate.dialect=

spring.jpa.hibernate.ddl-auto=

spring.profiles.active=

Требуется ввести информацию о БД:

spring.datasource.url=jdbc:postgresql://<IP address of your postgres server>/<the name of your database>

datasource.username и datasource.password - это данные для входа, которые используются для доступа к серверу. Имя пользователя по умолчанию - «postgres», а пароль по умолчанию - «password».

Нужно установить диалект. Для каждого типа базы данных существуют разные диалекты. Диалект сообщает Spring тип базы данных, к которой он будет подключен, и устанавливает правило для подключения к ней. Поскольку используется Postgres, нужно настроить его на использование диалекта Postgres:

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

Hibernate.ddl-auto определяет, как управляются данные. Если установлено значение «auto», Spring будет управлять данными, а затем передавать их в базу данных. Это самый быстрый способ настройки, но он не позволяет изменять варианты переноса и немного усложняет генерацию исходных данных. Если для параметра установлено значение «validate», можно вносить изменения в базу данных с помощью файлов .sql в дополнение к внесению изменений в базу данных через сам API. Команда для использования опции «validate»:

spring.jpa.hibernate.ddl-auto=validate

Spring.profiles.active позволяет изменять параметры между профилями «production», «development» и «test». Команда для использования профиля development:

spring.profiles.active=development

После ввода параметров файл finshed application.properties должен выглядеть:

spring.datasource.url=jdbc:postgresql://127.0.0.1:5432/testdb

spring.datasource.username=postgres

spring.datasource.password=shoppass

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

spring.jpa.hibernate.ddl-auto=validate

spring.profiles.active=development

**1.5 Лабораторная работа – Базовая настройка Java Spring**

Цель лабораторной работы: изучить принципы создания нового проекта Java Spring, провести базовую настройку и получить навыки работы с реляционной базой данных PostgreSQL.

Задание:

Выполнить пошагово следующие пункты задания:

1. Инициализируйте проект на сайте <https://start.spring.io/>.

2. Выставите настройки как на рис. 1.16.

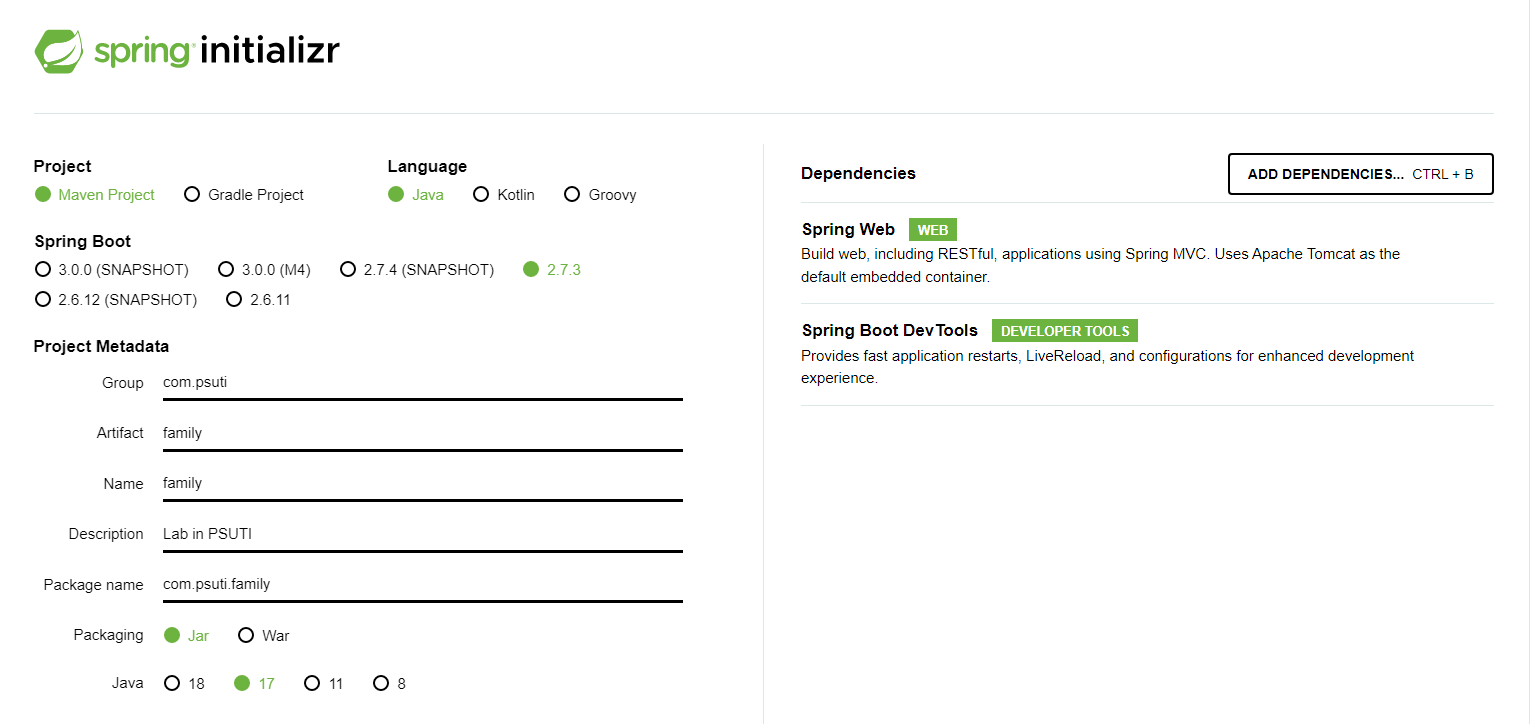


Рис. 1.16 – Исходные настройки

3. Вместо Artifact и Name впишите свою фамилию.

4. В зависимостях добавьте Spring Web (служит для работы REST) и Spring Boot DevTools (удобный инструмент для разработки).

5. Остальные зависимости укажите вручную в файле pom.xml (см. пример в п. 1.3).

6. Установите PostgreSQL.

7. Выполните первоначальную настройку и подготовку к работе c данной СУБД. В качестве имени БД используйте номер зачетной книжки, в качестве названия таблицы используйте свою фамилию, имя пользователя в БД используйте свое имя.

8. Произведите подключение к БД.

Контрольные вопросы:

Вопросы к защите лабораторной работы включают процесс установки и настройки по каждому пункту задания, а также теоретический материал к данной работе.

**Список использованных источников**

1. Учебное пособие по Spring Framework: начало работы со Spring [Электронный ресурс] / 2022. - Режим доступа: <https://bestprogrammer.ru>, свободный. - Загл. с экрана.
2. Java SE at a Glance [Электронный ресурс] / 2022. - Режим доступа: <https://www.oracle.com>, свободный. - Загл. с экрана.
3. Spring initializr [Электронный ресурс] / 2022. - Режим доступа: <https://start.spring.io>, свободный. - Загл. с экрана.
4. Установка PostgreSQL в Ubuntu [Электронный ресурс] / 2022. - Режим доступа: <https://losst.ru>, свободный. - Загл. с экрана.