

Aplicații necesare pentru desfășurarea probei practice de laborator din cadrul examenului la SD

Sunt necesare următoarele aplicații împreună cu o serie de pachete software suplimentare după cum urmează:

- IntelliJ IDEA Community Edition **cu plugin-ul pentru Kotlin**
- PyCharm Community Edition
- Python 3 (verificați cu comanda: **python --version**)
- Java Development Kit 8 (verificați atât *runtime*-ul, cât și compilatorul: **java** și **javac** - să fie la versiunea **1.8.x_xxx**)

Atenție la arhitectură! Versiunea corectă pe care trebuie să o descărcați și instalați este **amd64 (x86_64)**.

- **Observație:** JDK 8 este compatibil cu toate laboratoarele de SD.
- Postman
 - **opțional**, alternativele ar fi:
 - comanda **curl** din Linux - se poate instala pe Debian folosind comanda: **sudo apt install curl**
 - utilizarea unui browser (doar pentru cereri GET)
- server RabbitMQ: activare serviciu, activare *plugin* de gestionare - **rabbitmq_management**, configurare utilizator (**vezi laboratorul 5**)
- SQLite3 (și, opțional, **sqlitebrowser**)
- RxKotlin (**vezi laboratorul 7**)
- Micronaut (**vezi laboratorul 11**)
- dependențe Kotlin (se adaugă fie în **pom.xml** pentru proiecte Maven, fie în **build.gradle** pentru proiecte Gradle):
 - introduse în **laboratorul 3**:
 - **spring-boot-starter-parent**
(<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-parent>)
 - **spring-boot-maven-plugin**
(<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-maven-plugin>)
 - spring-boot-starter-web
(<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web>)
 - kotlin-maven-allopen
(<https://mvnrepository.com/artifact/org.jetbrains.kotlin/kotlin-maven-allopen>)
 - spring-boot-devtools
(<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools>)
 - **org.json.json** (<https://mvnrepository.com/artifact/org.json/json>)
 - introduse în **laboratorul 5**:
 - spring-boot-starter-amqp
(<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-amqp>)
 - introduse în **laboratorul 6**:
 - spring-boot-starter-jdbc

- (<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-jdbc>)
 - `sqlite-jdbc` (<https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc>)
- introduse în **laboratorul 7**:
 - `io.reactivex.rxjava3:rxkotlin:3.0.0`
(<https://mvnrepository.com/artifact/io.reactivex.rxjava3/rxkotlin>)
- introduse în **laboratorul 8**:
 - `maven-assembly-plugin` (necesar doar pentru **Maven**)
(<https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-assembly-plugin>)
 - * `docker` (+ abilități de încapsulare a microserviciilor în `docker`)
 - `com.github.johnrengelman.shadow` - necesar doar pentru **Gradle**
(<https://mvnrepository.com/artifact/com.github.johnrengelman.shadow/com.github.johnrengelman.shadow.gradle.plugin?repo=gradle-plugins>)
- introduse în **laboratorul 11**:
 - `log4j-api` (necesar doar pentru **Maven**)
(<https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api>)
 - `khttp` (<https://khttp.readthedocs.io/en/latest/>)
(<https://mvnrepository.com/artifact/khttp/khttp/1.0.0>)
- introduse în **laboratorul 13**:
 - `spark-core_2.12`
(https://mvnrepository.com/artifact/org.apache.spark/spark-core_2.12/2.4.5)
 - `spark-sql_2.12`
(https://mvnrepository.com/artifact/org.apache.spark/spark-sql_2.12/2.4.5)
 - `spark-streaming_2.12`
(https://mvnrepository.com/artifact/org.apache.spark/spark-streaming_2.12/2.4.5)
 - `mysql-connector-java` (<https://mvnrepository.com/artifact/mysql/mysql-connector-java>)
 - ~~`spark-streaming-kafka-0-10_2.12`~~
 - corutine introduse de la disciplina **Paradigme de Programare**:
 - `kotlinx-coroutines-core` (vezi laboratorul 10 de PP)
 - `kotlinx-coroutines-debug` (vezi laboratorul 10 de PP)
 - dependență suplimentară pentru serializare/deserializare **JSON**:
 - `kotlinx.serialization`
(<https://github.com/Kotlin/kotlinx.serialization#quick-example>)
- dependențe Python:
 - pachete necesare pentru medii virtuale / instalare de module (se instalează cu gestionarul de pachete al sistemului, de exemplu **apt**):
 - `python3-venv`
 - `python3-pip`
 - module ce trebuie instalate cu PIP:
 - ◆ introduse în **laboratorul 5**:
 - `retry`
 - `pika`
 - ◆ introduse în **laboratorul 6**:
 - `requests`

- ◆ introduse în **laboratorul 13**:
 - **pynput**
 - **pyspark**

Instalarea modului **kotlinx.serialization** pentru **Gradle**:

1. File -> New -> Project -> Gradle -> se bifează doar Kotlin/JVM -> Next -> Finish
2. Se modifică fișierul `build.gradle` în felul următor:

```
plugins {
    id 'org.jetbrains.kotlin.jvm' version '1.3.72'
    id 'org.jetbrains.kotlin.plugin.serialization' version '1.3.72'
}

group 'com.sd.exam'
version '1.0.0'

repositories {
    mavenCentral()
    jcenter()
}

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
    implementation "org.jetbrains.kotlinx:kotlinx-serialization-runtime:0.20.0"
}

compileKotlin {
    kotlinOptions.jvmTarget = "1.8"
}

compileTestKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
```

3. **Utilizarea modului `kotlinx.serialization`**:
<https://github.com/Kotlin/kotlinx.serialization#quick-example>

Instalarea modului **kotlinx.serialization** pentru **Maven**:

1. File -> New -> Project -> Maven -> se bifează „create from archetype” și se selectează opțiunea **org.jetbrains.kotlin:kotlin-archetype-jvm:1.3.72** -> Next -> Next -> Finish
2. Se modifică fișierul `pom.xml` în felul următor:

```
<?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 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.sd.exam</groupId>
  <artifactId>MavenSerialization</artifactId>
  <version>1.0.0</version>
  <packaging>jar</packaging>
  <name>com.sd.exam MavenSerialization</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <kotlin.version>1.3.72</kotlin.version>
    <kotlin.code.style>official</kotlin.code.style>
    <junit.version>4.12</junit.version>
    <serialization.version>0.20.0</serialization.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.jetbrains.kotlin</groupId>
      <artifactId>kotlin-stdlib</artifactId>
      <version>${kotlin.version}</version>
    </dependency>
    <dependency>
      <groupId>org.jetbrains.kotlin</groupId>
      <artifactId>kotlin-test-junit</artifactId>
      <version>${kotlin.version}</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.jetbrains.kotlinx</groupId>
      <artifactId>kotlinx-serialization-runtime</artifactId>
      <version>${serialization.version}</version>
    </dependency>
  </dependencies>

  <build>
    <sourceDirectory>src/main/kotlin</sourceDirectory>
    <testSourceDirectory>src/test/kotlin</testSourceDirectory>
    <plugins>
      <plugin>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-maven-plugin</artifactId>
        <version>${kotlin.version}</version>
        <executions>
          <execution>
            <id>compile</id>
            <phase>compile</phase>
            <goals>
              <goal>compile</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <compilerPlugins>
```

```
        <plugin>kotlinx-serialization</plugin>
    </compilerPlugins>
</configuration>
</dependencies>
<dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-maven-serialization</artifactId>
    <version>${kotlin.version}</version>
</dependency>
</dependencies>
</plugin>
</plugins>
</build>
</project>
```

3. Maven -> <Project Name> -> Lifecycle -> clean + compile

4. Utilizarea modulului kotlinx.serialization:

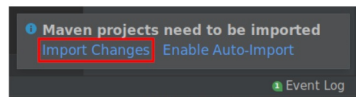
<https://github.com/Kotlin/kotlinx.serialization#quick-example>

Observație referitoare la importarea modificărilor aduse fișierelor de configurare pentru gestionarea de proiect (Maven - pom.xml / Gradle - build.gradle):

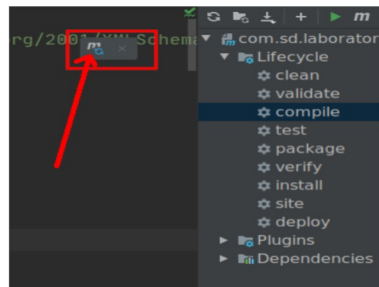
Începând de la versiunea de **IntelliJ 2020.x**, **țineți cont că s-a modificat modalitatea de importare a modificărilor făcute în proiect**, conform figurii următoare:

Maven

IntelliJ ≤ 2019.x

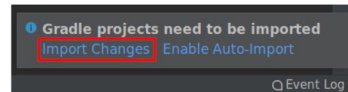


IntelliJ ≥ 2020.x



Gradle

IntelliJ ≤ 2019.x



IntelliJ ≥ 2020.x

