



# FUNDAMENTAL PROGRAMMING TECHNIQUES

## Laboratory Resources

2025



## Contents

Contents .....	2
1 Java .....	3
1.1 Java JDK .....	3
1.2 Set the JAVA_HOME variable .....	5
2. IntelliJ IDEA .....	6
3. Database Server .....	9
4 Git and Maven Projects.....	12
4.1 Git installation .....	12
4.2 Create Account on GitLab.....	13
Inside the group, you can create your own projects for different applications of the PT lab. Click on <i>Create new project</i> . .....	14
4.3 Basic Instructions .....	15
4.3.1 Create a project from scratch.....	15
4.3.2 Update the project.....	17
4.3.3. Create and work with a new branch .....	18
4.3.4. Getting git to work with a proxy server.....	20
4.3.5. Getting MAVEN to work with a proxy server .....	20



# TECHNICAL UNIVERSITY

OF CLUJ-NAPOCA, ROMANIA

## 1 Java

### 1.1 Java JDK

- 1) Access the next link:

<https://www.oracle.com/java/technologies/downloads/>

JDK 23   JDK 21   GraalVM for JDK 23   GraalVM for JDK 21

#### Java SE Development Kit 21.0.6 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions](#) (NFTC).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN](#) beyond the [limited free grants](#) of the OTN license will [require a fee](#).

Linux   macOS   Windows

Product/file description	File size	Download
x64 Compressed Archive	185.92 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip</a> (sha256)
x64 Installer	164.31 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe</a> (sha256)
x64 MSI Installer	163.06 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi</a> (sha256)

- 2) Click on the link which corresponds to your version of the Operating System. In the example the version which is used corresponds to Windows x64 and the file is named [https://download.oracle.com/java/21/latest/jdk-21\\_windows-x64\\_bin.exe](https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe).
- 3) Click on *jdk-21\_windows-x64\_bin.exe*. The window below will appear. Click on the *Next* button.



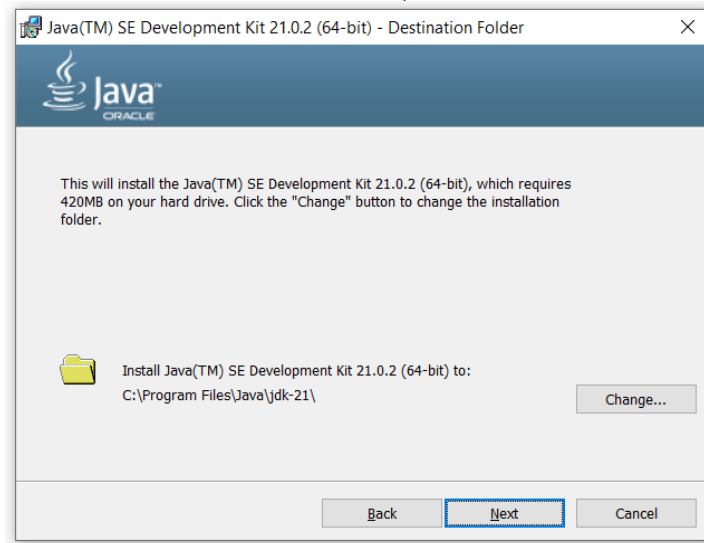
- 4) You will be asked where you want to install Java. Use the default location and click Next.



---

**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA, ROMANIA

---



- 5) After the installation is successfully completed the window below will be displayed. Click Close.





---

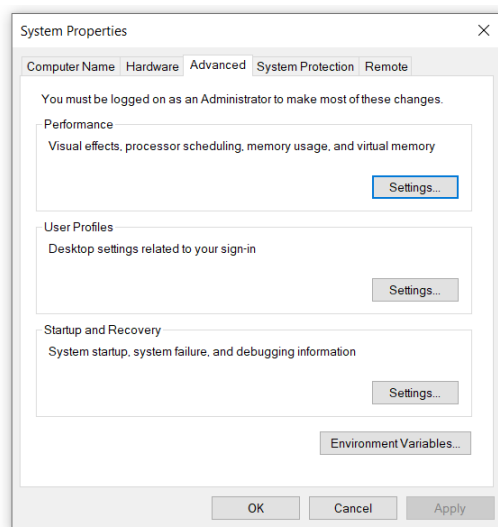
**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA, ROMANIA

---

## 1.2 Set the JAVA\_HOME variable

*Note: the steps may vary according to the Windows version installed on the computer*

- 1) Write in the search bar: “Edit the system environment variables” and click on the suggested option. The window below will be displayed.



- 2) Click on *Environment Variables*.
- 3) Under *System Variables* click *New*.
- 4) In the text field associated with the name of the variable insert `JAVA_HOME` and in the field associated with the value of the variable insert `C:\Program Files\Java\java_version;`.
- 5) Click *OK*.

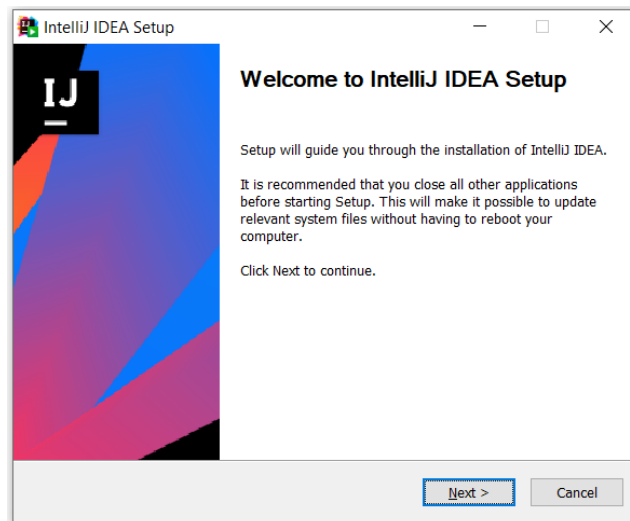


## 2. IntelliJ IDEA

We recommend using IntelliJ IDEA as the IDE for developing your applications during the laboratories. As a student, you can benefit of a free educational license for IntelliJ IDEA, by applying here <https://www.jetbrains.com/community/education/#students> -> Click on *Apply now* -> Fill in the requested information and click on *Apply for free products*. After obtaining the license, download the latest version of IntelliJ IDEA. Click on the downloaded file. You will be asked whether you allow the program to make changes to this computer -> Click Yes.

Then, follow the steps below:

- 1) Click next



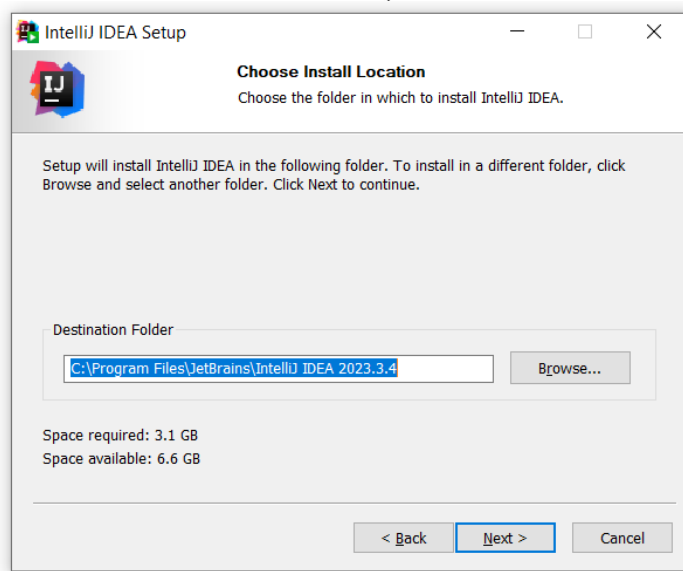
- 2) You will be asked where you want to install IntelliJ IDEA. Use the default location and click Next.



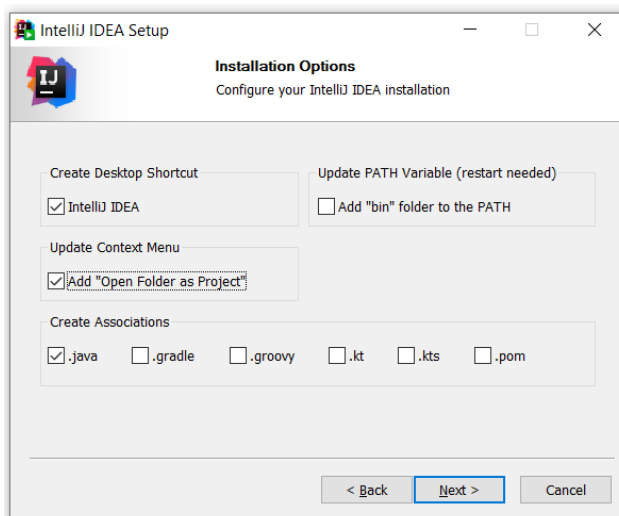
---

**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA, ROMANIA

---



3) Configure the installation options (see an example in the image below) and click Next.



4) Click Install.

5) After the installation completes click on *Finish*.

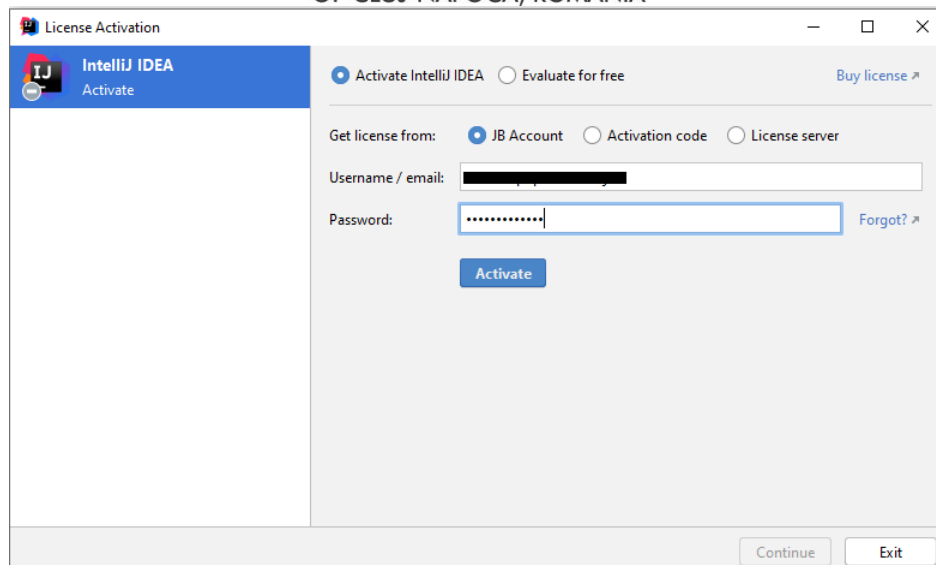
6) Run IntelliJ IDEA. When you first run IntelliJ IDEA after the installation completes you will have to activate it by inserting the username/email and password you used when creating the account for obtaining the students license. Click *Activate*.



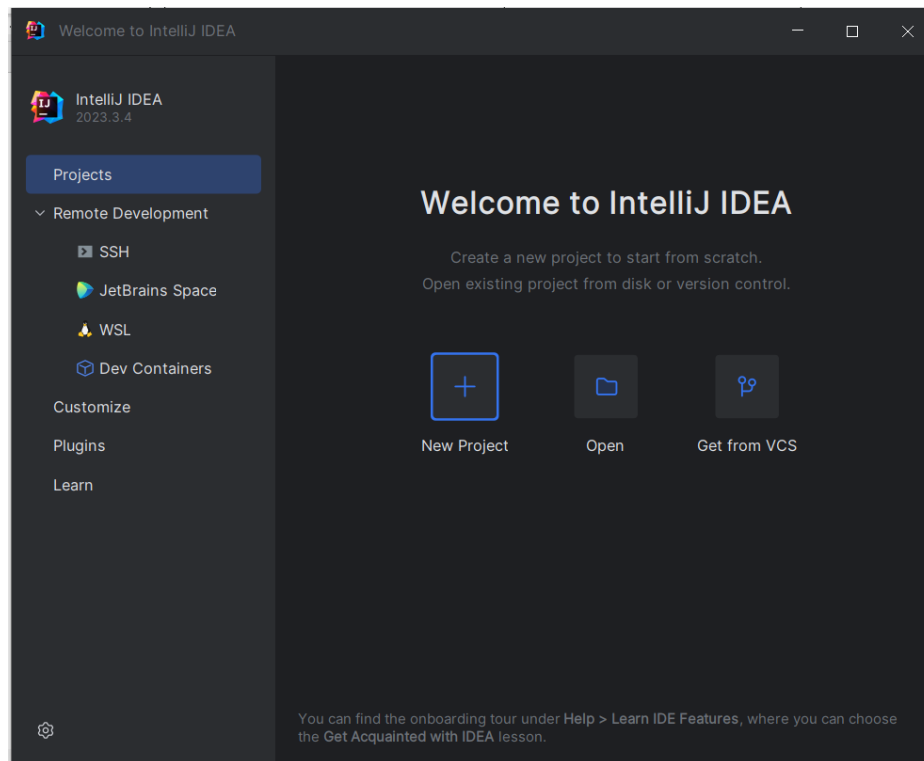
---

**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA, ROMANIA

---



7) After the activation completes, the IDE will open.







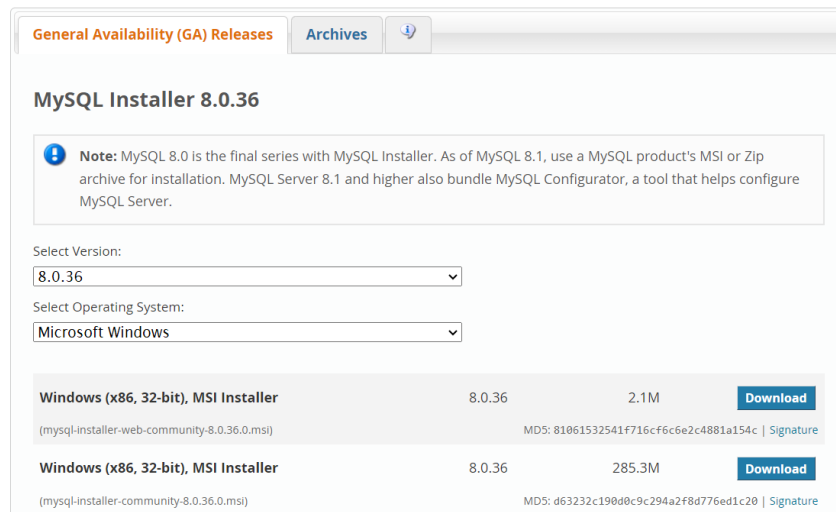
### 3. Database Server

A database server must be installed on the local machine. A MySQL server can be used to run locally the projects. For installing the MySQL server follow the next steps:

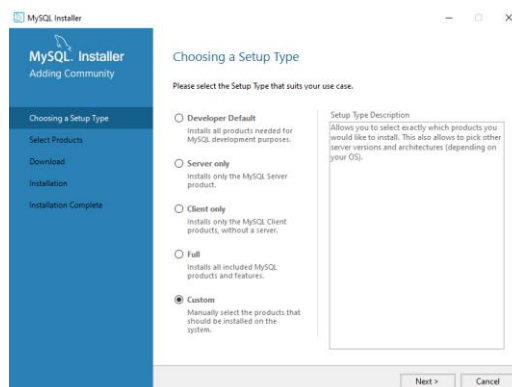
- 1) Click on the next link to download the latest version:  
<https://dev.mysql.com/downloads/windows/installer/>.

#### MySQL Community Downloads

MySQL Installer



- 2) Click on the second Download button.
- 3) Click on *No thanks, just start my download*.
- 4) Click on the downloaded file.
- 5) You will be asked whether you allow the program to make changes to this computer -> Click *Yes*.
- 6) You will be asked to select the Setup Type that suits your use case. Select *Custom* and click *Next*.

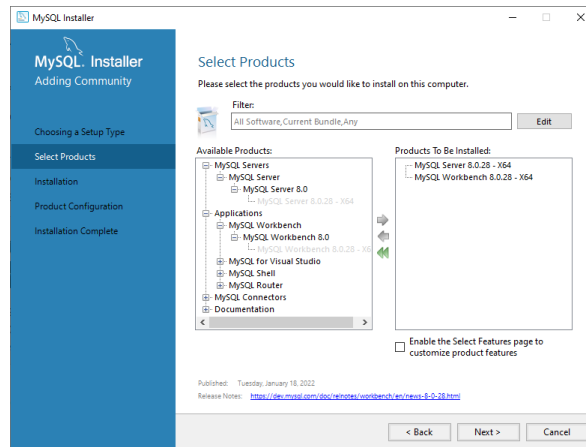




## TECHNICAL UNIVERSITY

OF CLUJ-NAPOCA, ROMANIA

- 7) You will be redirected to “Select Products and Features”. Select “MySQL Server *version*” and “MySQL Workbench *version*” and click *Next*.

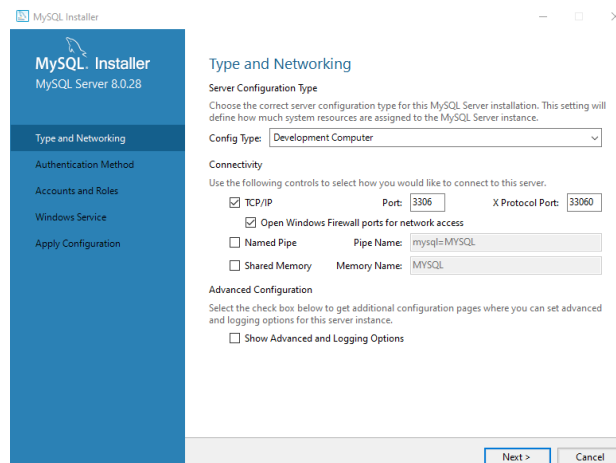


- 8) Click *Next*.

- 9) Click *Execute*.

- 10) Click *Next*.

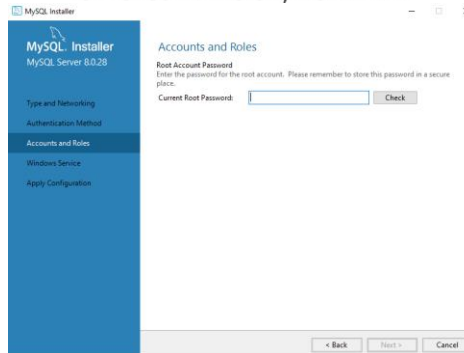
- 11) Click *Next* and follow the steps for the configuration of the MySQL Server. At this stage pay attention to the port on which the MySQL server is running on (i.e., 3306 is the default port).



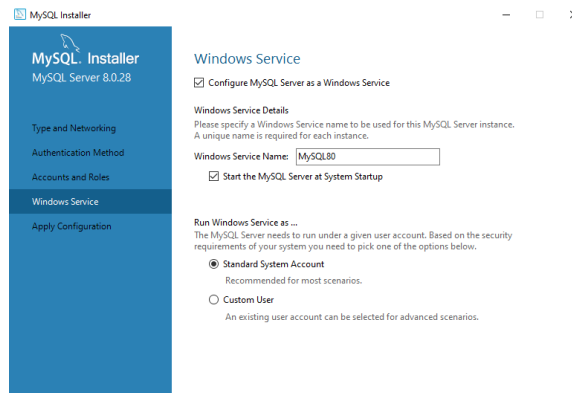
- Click *Next*
- Select the Authentication Mode (select *Use Strong Password Encryption for Authentication*) and click *Next*.
- Set the password for the root account – this must be set by you and make sure you remember it as it will be used for further connections to the MySQL server. After inserting the password click on the *Check* button.



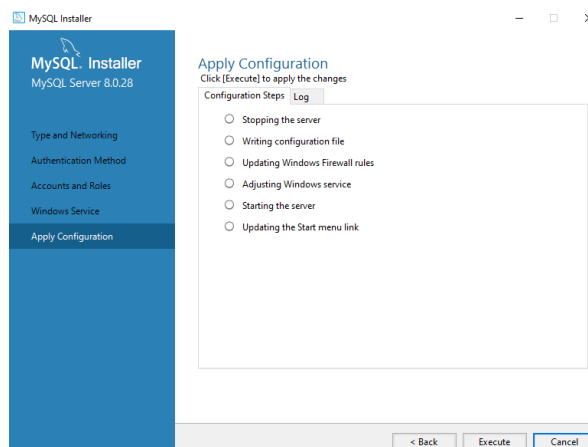
## TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, ROMANIA



- Click *Next*



- Click *Next* and then *Execute*



12) Click Finish.

**Note:** in case you uninstall MySQL, follow the steps presented [here](#) to completely uninstall it.



## 4 Git and Maven Projects

Modern software projects development requires effort from large teams of developers that have to collaborate in order to integrate their work and create the final product. Coordinating and tracking the changes between multiple source files is a difficult task, thus an automatic tool was developed in 2005, initially for the development of the Linux Kernel, and, since then, it has penetrated all levels of software development. The tool is a version control system (VCS), named GIT, which tracks changes of computer files and helps coordinating several people who work on those files.

Furthermore, large applications often encounter problems with the software build settings as well as with the dependency description. To address these problems, another automatic tool was created, MAVEN that defines conventions for the build procedure and uses an XML file to describe the software project, dependencies, external modules, components and plug-ins.

### 4.1 Git installation

- 1) Click on <https://git-scm.com/downloads>.
- 2) Select your operating system.



- 3) If you select Windows, a file called *Git-version-64-bit.exe* should be downloaded. In the case you select another operating system or if your system is on 32 bits then a file with a similar name should be downloaded.
- 4) Click on this file and follow the default installation guidelines, except for the step where you are asked which terminal emulator you want to use. Select the second option.



## TECHNICAL UNIVERSITY

OF CLUJ-NAPOCA, ROMANIA

### 4.2 Create Account on GitLab

- 1) Click on <https://about.gitlab.com/>.
- 2) In the right corner, click on Register. You will be asked to introduce your personal information. Or, if you already have an account, just *Sign In*.

- 3) In the next window, you can choose the role of “Software Developer”. In the next checkbox, you can choose the “Just me” option.
- 4) Following up, you will have to create a group. The name of the **PRIVATE** group must be of the format: *PT2025\_GroupNumber\_LastName\_FirstName*  
(e.g. *PT2025\_30441\_Popescu\_Ioan*)
- 5) For now, do not invite any other teammates to have access to the group, as we will do this later.



## TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, ROMANIA

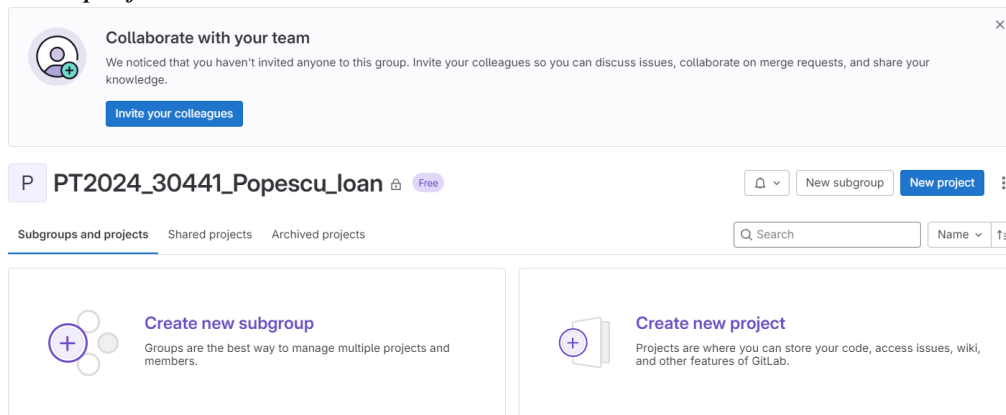
6) Click *Create group*.

If you already had an account and just signed in, you must create the group we have just talked about. Go to: *Groups* → *Your Groups* → *New Group* then create a new **PRIVATE** group with the format: *PT2025\_GroupNumber\_LastName\_FirstName*

(e.g. *PT2025\_30441\_Popescu\_Ioan*)

7) Now you must give access to your group, to the lab assistants. On your Group page, go to: *Manage* → *Members* → *Invite Member* → and offer **Maintainer** rights for the user: utcn\_dsrl.

Inside the group, you can create your own projects for different applications of the PT lab. Click on *Create new project*.



Click on *Create blank project*. Remember to keep the same naming conventions for the projects.

PT2024\_30441\_Popescu Ioan / New project / Create blank project

### Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

**Project URL**

**Project slug**

**Project deployment target (optional)**

**Visibility Level** [?](#)

☒ Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

**Project Configuration**

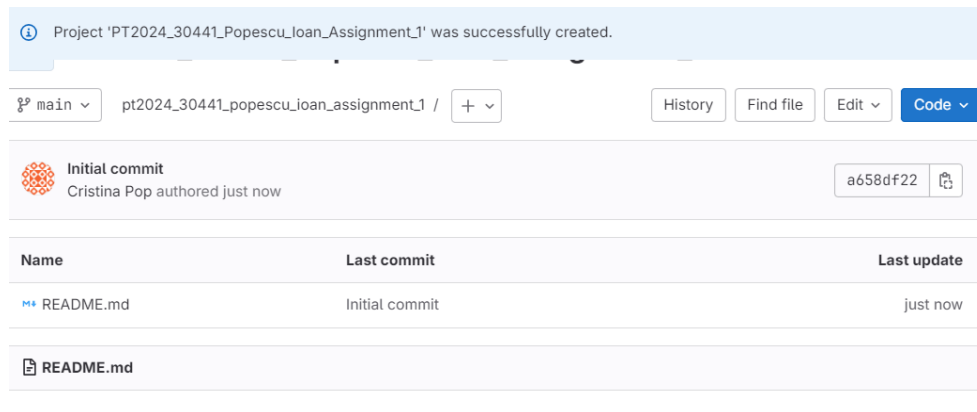
☒ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)



## TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, ROMANIA

Click on *Create project*.

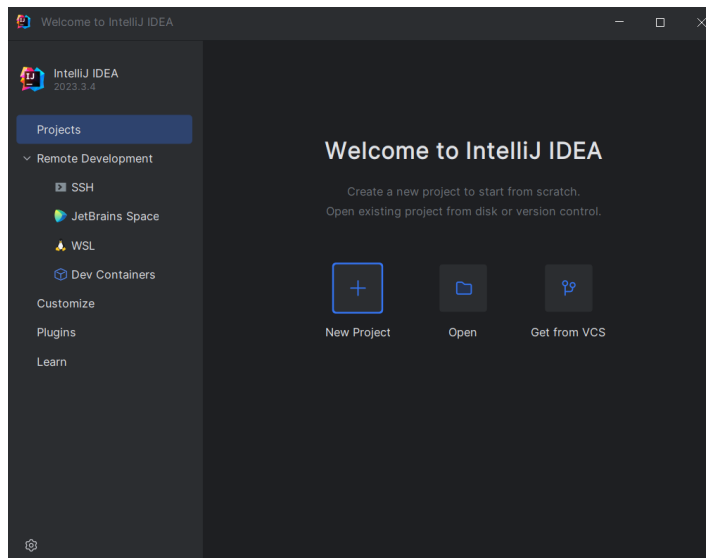


### 4.3 Basic Instructions

#### 4.3.1 Create a project from scratch

- 1) Create the folder *PT2025\_GroupNumber\_LastName\_FirstName* on *D:\*.

Inside this folder create a new project. Open IntelliJ, click on *New Project*.



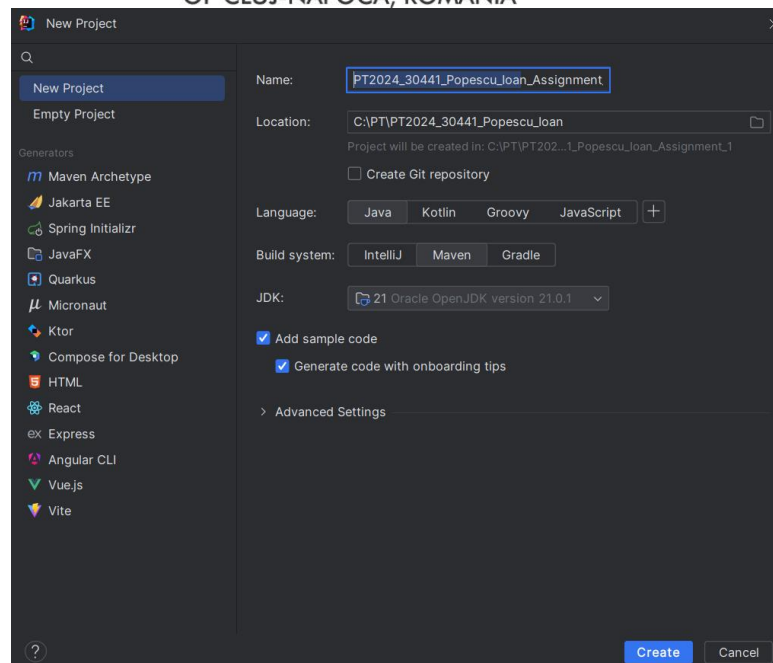
Then choose *Maven* from the list from the left, choose the appropriate Java SDK, then click on *Create*. Instead of using the default location suggested, use this one:

*PT2025\_GroupNumber\_LastName\_FirstName*

*Do not forget to also choose the appropriate name for your new Project, most likely using the format we requested (unless it is a test project, not for one of your assignments).*



## TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, ROMANIA



- 2) Click *Create*.
- 3) Now, you must create a *.gitignore* file in the project if it is not already created, which tells git which files to ignore when committing and pushing to your remote projects. You don't want unnecessary files, such as IDE configuration files, to be pushed, because they are strictly relevant for your local system. Just create a file named *“.gitignore”* and write the following lines:

```

1  /target/
2  !.mvn/wrapper/maven-wrapper.jar
3
4  ### STS ###
5  .apt_generated
6  .classpath
7  .factorypath
8  .project
9  .settings
10 .springBeans
11 .sts4-cache
12
13 ### IntelliJ IDEA ###
14 .idea
15 *.iws
16 *.iml
17 *.ipr
18

```

- 4) Right click on the folder *PT2025\_GroupNumber\_LastName\_FirstName\_Assignment\_1* and click *Git Bash Here*.
- 5) Write the next commands:



**a) git init**

*Note: the command will create a local repository with a default main branch. Notice that after executing this command, a hidden .git folder will be created inside the current folder representing the repository where git stores all necessary files. From now on, it will be possible to track all the changes that will be performed to the files from the original folder. The original folder is considered to be the working directory, while the .git folder is referred as the repository that tracks the made changes (for more details check this [link](#)).*

**b) git remote add origin**

[https://gitlab.com/group\\_name/repository\\_name.git](https://gitlab.com/group_name/repository_name.git)

For example:

`git remote add origin`

[https://gitlab.com/pt2025\\_30441\\_popescu\\_ioan/pt2025\\_30441\\_popescu\\_ioan\\_assignment\\_1.git](https://gitlab.com/pt2025_30441_popescu_ioan/pt2025_30441_popescu_ioan_assignment_1.git)

*Note: this command will connect the local repository created using the “git init” command with the remote repository’s origin (for more details check this [link](#)).*

**c) git add .**

*Note: the command will mark any changes that you have made to your project files (e.g. creating/modifying/deleting files) as staged so that they can be included in the next commit (for more details check this [link](#)).*

**d) git commit -a -m “initial commit”**

*Note: the command commits any files you have added with the git add command and commits any files you have changed since then – at this step the changes are saved only locally. An explanation message is given in order to document what has been added/changed (for more details check this [link](#)).*

**e) git branch -M main****f) git push -uf origin main**

*Note: the command sends the committed changes to your remote repository (for more details check this [link](#)).*

**4.3.2 Update the project**

- 1) Create a new class named *App* in the same package as the class *Main*.
- 2) Navigate to folder *PT2025\_GroupNumber\_LastName\_FirstName\_Assignment\_Number*, right click and select *Git Bash Here*
- 3) Insert the next commands:
  - a) git add .**
  - b) git commit -a -m “add new class”**
  - c) git pull origin main**



---

**TECHNICAL UNIVERSITY**

---

OF CLUJ-NAPOCA, ROMANIA

*Note: the command fetches and merges changes on the remote server to your working directory (for more details check this [link](#)).*

**d) git push -u origin main**

4) You can always see the modification that were not committed yet by using:

**a) git status****4.3.3. Create and work with a new branch**

The real value of working with git, is the power of *branches*. They allow multiple developers to work simultaneously on the same project, on different features, and then to merge all the new changes in the main branch.

**1) Create a branch production**

The first step towards working with branches, is to create a new branch. When you create a new branch, it will automatically be initialized with the currently existent code. Then, while inside that branch, all changes will be added only on that branch.

To create a new branch:

- pull all the changes from the remote project, to be up to date:
  - *git pull*
- create the branch on your local machine and switch directly to that branch:
  - *git checkout -b <branch-name>*
- push the newly created branch to the remote repository:
  - *git push origin <branch\_name>*

An example to create the branch production is the following:

```
#git pull
#git checkout -b production
#git push origin production
```

**2) switch between branches**

When working with multiple branches, it is important to keep track of all the available branches, and to always know on which branch you currently are.

- To bring locally meta-data information about existing branches:
  - *git fetch --all*
- In order to see all existent branches:




---

**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA, ROMANIA

---

- *git branch -a*
- In order to see on which branch you currently are:
  - *git status*
- *in order to switch from a branch to another, use the above-mentioned command:*
  - *git checkout <branch-name>*
    - *if the branch with that name is already existent, it will just switch to that one, instead of creating a new one*

*You can try to create a new branch, make some small change, then switch back to the main branch, and see that change is not present in the main branch.*

### 3) commit changes to new branch

When making changes on a new branch, you must always commit and push them to the remote branch, just like working on main.

- First, make sure you are on the right branch:
  - *git status*
- Repeat the same process as if you were working with main. However, pay attention to the names:
  - *git add .*
  - *git commit -m "commit message"*
  - *git push -u origin <branch-name>*
- And now, your remote branch <branch-name> contains all the changes you have pushed.

### 4) merge branch with main

An important step when working with branches, is to always keep the main branch up to date with the latest **working and functional** code from your other branches. Merging two branches, as the name suggests, is the process of merging the code from two branches. If the branches contain changes on different parts of the code, the merge process will work instantly. If both branches contain changes on the same parts of code, git will require you to solve the conflicts: from the two modifications, you must choose the one which you want to remain in the final version.

**DO NOT FORGET:** do not merge code which is not working properly, or which is not tested, into main. The main branch must always contain the latest functional version of your project.

In order to merge two branches:

- *git merge <branch-with-new-changes> <branch-to-be-updated>*



## TECHNICAL UNIVERSITY

OF CLUJ-NAPOCA, ROMANIA

For a more in-depth explanation of branches and how they can be manipulated to serve your needs, we suggest checking the following tutorial:

<https://git-scm.com/book/en/v2/Git-Brambling-Basic-Brambling-and-Merging>

### 4.3.4. Getting git to work with a proxy server

- 1) In the UTCN laboratories you need to set the proxy server to use GIT bash
- 2) Open Git Bash
- 3) Insert the following commands:
  - a) `git config --global http.proxy http://proxy.utcluj.ro:3128`
  - b) `git config --global --get http.proxy`
- 4) In order to unset the proxy, use the following command:
  - a) `git config --global --unset http.proxy`

### 4.3.5. Getting MAVEN to work with a proxy server

- 1) In the UTCN laboratories you need to set the proxy server in order to use MAVEN projects
- 2) Go to Windows Explorer-> Drive C-> Users -> *Your User* -> .m2
- 3) Create the folder **conf**
- 4) Go to conf folder and create the file **settings.xml** with the following content:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository/>
  <interactiveMode/>
  <usePluginRegistry/>
  <offline/>
  <pluginGroups/>
  <servers/>
  <mirrors/>
  <proxies>
    <proxy>
      <id>myproxy</id>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.utcluj.ro</host>
      <port>3128</port>
      <username></username>
      <password></password>
```



---

**TECHNICAL UNIVERSITY**

---

OF CLUJ-NAPOCA, ROMANIA

```
<nonProxyHosts>localhost,127.0.0.1</nonProxyHosts>
</proxy>
</proxies>
<profiles/>
<activeProfiles/>
</settings>
```

- 5) Go back to folder **.m2**
- 6) Delete the folder **repository**
- 7) For Eclipse
  - a) Open **Eclipse**
  - b) Go to **Window->|Preferences->|Maven->|User Settings**
  - c) At the **User Settings** tab browse for the **settings.xml** file created at step 4
  - d) Click **Apply** and **OK**
  - e) Go on the project, right click and go to **Maven->Update Project**
- 8) For IntelliJ IDEA
  - a) Open IntelliJ IDEA
  - b) Go to **File->|Settings->|Build, Execution, Deployment->|Build Tools->|Maven**
  - c) In the **User Settings file** field browse for the **settings.xml** file created at step 4.
  - d) Click **Apply** and **OK**