# Project Guidelines

## 1 About this Document

This document provides students with all information regarding the project organization and deadlines. It is ***important*** to read it carefully since it includes the answers to the majority of the questions regarding the project.

## 2 Purpose

The purpose of the project is to define and implement a distributed application, using communication abstractions and algorithms presented in the course. In order to do so, you need to create an application that you are free to choose its subject (see section 7.1 for examples of topics of previous years projects). The application must be one of theses two types:

- **Client-server:** the entities roles are defined in advance;
- **Peer-to-peer:** no predefined client or server role is assumed for the collaborating entities.

You should use RMI and/or sockets to implement your application. Furthermore, once you have seen the related concepts in the lecture (i.e., in mid-November), we ask you to add a mobile client (iPhone) to your application.

## 3 Software/Hardware Requirements

For iPhone programming, students need Mac computers with *Xcode (6.X )* or later versions installed on it. For this purpose, UNIL students and more precisely students of masters of information systems (ISI) could use ***Mac computers*** in FAME room (i.e., room Internef 261 at UNIL) since they could have access to FAME room everyday of the week (with UNIL campus card). For EPFL students, we could only guarantee access to FAME room during the exercise sessions, therefore they need to make sure by themselves that they can use a Mac machine with required software installed on it for the iPhone programming part of the project. For the Java programming parts of the project, we use *Netbeans IDE*. As the version control system, we use *Git*. As the Git clients, we use *Netbeans built-in git client* for Java code and *Xcode built-in git client* for Objective-C code. For more information about Git, please read Section 8 (Appendix).

# 4 Prerequisite

The prerequisite of this course is the course *Object Oriented Programming (OOP)* (see: `http://doplab.unil.ch/?q=oop`). So, in order to perform the project, you need a good knowledge of Java programming. You also need an average knowledge of Objective-C or a strong will to learn Objective-C and iPhone programming in a short time duration (about 2 or 3 weeks).

# 5 Gitlab Account

During the project, you will use your gitlab username/password to connect to your project repositories on `https://gitlab.unil.ch/`. The repositories are created by the teaching staff for your group at the beginning of step 2 of your project. Depending on you university, there are two types of gitlab accounts:

- **Students who have a UNIL email address:** if you have an email address at UNIL, your gitlab username/password are the same as the one's used to connect to your UNIL mailbox at `https://owa.unil.ch/owa/`.
- **Other students (including EPFL students):** you will create your gitlab account at the beginning of the project using the "New user? Create an account" window of gitlab that will be enabled at that time (see Section 6.1.1).

# 6 Project Steps

## 6.1 Step 1 (October 20 - November 10)

In this step, you first form your groups and then define your projects specifications.
In order to define your project specification, you can download the file *Project Specification Template.rtf*, which can be found on the website of the course under "Project" page. You can use the information in this file as an example and write your project specification. At the end of this step, each group presents its specification. You can find example(s) of previous year presentations under the "Project" page.

### 6.1.1 Group Formation

Each group should be composed of 4 or 5 members. Note that at this stage, we do not accept the groups that are composed of less than 4 students. ***If you are less than 4 students in your group or if you are alone and you are not able to find any group, you must send an e-mail***

***to `vaibhav.kulkarni@unil.ch` and `arielle.moro@unil.ch` before the deadline, i.e., October 19 at 23:59***. We then try to make groups of 4 or 5 students out of incomplete groups. Thus, in order to create a group, each group should do the following:

1. Each member of the group should activate her/his gitlab account ( you can skip this step if you have already activated your account on gitlab). In order to do so, each member should open: `https://gitlab.unil.ch/` in a web browser. Then, there exist two cases:

   (a) *the member has a UNIL email address*: in this case, in the window "Existing user? Sign in" and under "LDAP" option, the member should login with username and password that she/he uses to connect to her/his UNIL mailbox at https://owa.unil. ch/owa/.

   (b) *the member does not have a UNIL email address*: in this case, she/he should fill the fields in the window "New user? Create an account". In this case, the field "Name" should be filled with the complete last name of the user and the field "Email" with her/his academic email address. Once the user click the Sign-up button, she/he will receive a confirmation email with a link to access her/his account.

   Once logged in, you can close the browser window. Note that ***it is very important that each group member does this process***. In fact, it enables us to create your group Git repositories on gitlab. You will use your group Git repositories from Step 2 of the project (i.e., after November 10). ***Please do not create any project or group for this course on gitlab by yourself. The teaching staff will create repositories for your group and send you the information regarding your repositories at the beginning of Step 2.*** Note that you should not use your Git repositories until the teaching staff announce the information regarding your repositories.

2. Once the above (activation of an account on gitlab) is done for all group members; each group should send to `vaibhav.kulkarni@unil.ch` and `arielle.moro@unil.ch` an email until **October 19 at 23:59**, and announces its group formation. Of course, one email per group is sufficient. Your email should respect the following format:

   - **Email title:** ids16group
   - **Email content:** you list your group members. For each member you should indicate: (first name, last name, academic email address). If you are less than 4 students in your group or if you are alone and you are not able to find any group, please indicate shortly this problem in your email.

Note that your emails only will be taken into consideration if they ***respect the above mentioned format*** and if they are sent ***before the deadline***.

### 6.1.2 Presentation of Project Specification

On **November 10** each group has to do a short presentation explaining its project specification. The exact time and place for each group presentation will be announced via email a few days before this date. You should send the slides of your presentation (in .pdf format) and your *Project Template.rtf* (filled with your project specification) until **November 9 at 23:59** to `vaibhav.kulkarni@unil.ch` and `arielle.moro@unil.ch`.
The presentations are public (i.e., they take place in front of the whole class). The presentations can be in English or French. All group members should be present during the presentation of their group. In the case that a member is absent, she/he should send us an email explaining the reason of the absence. It is preferable that each group member presents a part of the presentation (1 or 2 slides or at least a part of a slide). Note that if you take this course, we assume that you keep free your Thursday afternoons for this course. Please do not send us an email and request to change the schedule. We can not change the schedule of the presentations according to the personal availability of the students. Each presentation will last 15 minutes and comprise two parts:

- **First part(about 10 min.):** presenting the slides. The slides should contain:
  - Title and context of your project: you mainly describe your motivation to define a new application
  - Goal and a short description of your application
  - Architecture of your application: client-server/peer-to-peer
  - Main building blocks of your application: e.g., Java Server, Java client, iPhone client,...
  - List of functionalities provided by each building block (including the iPhone client)
  - Communication between building blocks: RMI/Sockets
  - Work distribution among group members: you should precise which part of the application will be developed by which member.
- **Second part (about 5 min.):** Questions & Answers.

## 6.2 Step 2 (November 10 - December 22)

In this step, you should implement your application according to the project specification that you defined in Step 1 and that are validated by us. At the end of this step, you present your work in a final presentation.
At the beginning of this step, you start to use your group Git repositories. The information about your group Git repositories will be sent to you by the teaching staff at the beginning of Step 2. In addition, you need to read Section 8 (Appendix) to know how you can use your group Git repositories for developing Java or Objective C applications.

### 6.2.1 Final Presentation

The deadline to submit your code and your slides is fixed for **December 21 at 23:59**. Your code (Java code as well as the objective-C code) and your slides should be found on the repositories of your group on `https://gitlab.unil.ch/` before the deadline. The slides should be in pdf format. You can create a folder called "docs" in your Java project and place the .pdf format of your slides in that folder. Note that we only check your Git repositories on `https://gitlab.unil.ch/` and on no other personal repositories. Furthermore, **you do not need to provide a report for the project**. Finally, we advise you to comment your code.

The final presentations will take place on **December 22** in FAME (Internef 261). The exact time for each group presentation will be announced via email a few days before this date. The presentations are public (i.e., they take place in front of the whole class). The presentations can be in English or French. All group members should be present during the presentation of their group. In the case that a member is absent, she/he should send us an email explaining the reason of the absence. It is preferable that each group member presents a part of the presentation (1 or 2 slides or at least a part of a slide). Note that if you take this course, we assume that you keep free your Thursday afternoons for this course. Please do not send us an email and request to change the schedule. We can not change the schedule of the presentations according to the personal availability of the students. Each presentation will last 15 minutes and be made up of three parts:

- **First part(about 5 min.):** A demo of your application. The demo can be done on your laptops or one of the FAME (Internef 261) machines. However, if you are an EPFL student we advise you to use your laptops for demos because your access to FAME room is only guaranteed during the exercise sessions. Note that for demos, all elements of your application ( e.g., server, Java client, iPhone client, etc...) can run on the same machine (local host).

- **Second part(about 5 min.):** Presentation consisting of:
  - Architecture of your application: client-server/peer-to-peer
  - Main building blocks of your application: e.g., Java Server, Java client, iphone client,...
  - Communication between building blocks: e.g., Java RMI, Socket programming.
  - Work distribution among group members.
- **Third part(about 5 min.):** Questions & Answers.

Some advice for your final presentations: (1) Do not repeat the demo in slides by using screen shots; (2) Get prepared as your presentation time is limited.

# 7 Miscellaneous

## 7.1 Previous Years Project Topics

Below you can find the list of previous years projects topics:

- Multi-user image gallery
- Distributed "Connect four" game (or jeu "puissance 4")
- A distributed chat application
- Social movie network: a distributed application for sharing movie critics between friends

This list gives you an example of the kind of the project that you can define. You are free to define your topic or choose a similar topic as a previous year project. Keep in mind that in any case, your application should have a distributed architecture (client-server or peer-to-peer).

# 8 Appendix

## 8.1 Git

Throughout this project, you should use Git as your version control system. A version control system manages all changes occurring to a software over time, that is, as the project is worked on and developed. Thus, a version control system allows more than one person to work on a single project. It also provides a complete history of all files, which allows the project to be backtracked and comparisons made with previous versions.

Git can be used in the following modes (for more information see `https://www.atlassian.com/git/workflows`):

1. Centralized workflow,
2. Feature branch workflow,
3. Gitflow workflow,
4. Forking workflow.

One of the advantages of using Git over other version control systems such as subversion(SVN), is that Git enables a user to save locally all the work progress and a history of it. In addition, as already described, we can use Git in different modes depending on our way of working. to know more about how Git works and all Git commands, you can check `https://www.atlassian.com/git/tutorial`.

In this project, you will use Netbeans built-in Git client and Xcode built-in Git client. In particular, you will use Netbeans built-in Git client while developing your Java projects and Xcode built-in Git client for the iPhone programming (your Objective-C program).

## 8.2 Gitlab web interface

For this project, you will use the Git repositories which are created for your group on `https://gitlab.unil.ch/`. ***Please note that you should not create any project or group for this course on gitlab by yourself. The teaching staff will create the repositories for your group and send you the information regarding your repositories at the beginning of Step 2 of the project***. You can access the web interface of gitlab using a web browser. Once logged in with your with gitlab username and password (see Section 5 if you do not know what are your gitlab username/password), on the right hand and under the "Projects" thumbnail you can see a window containing the list of all repositories that you have access to (see Figure 1).

Thus, for this course you have access to the following repositories:

- "ids16gX-java" for Java code with Netbeans tool, where X is the number of your group which will be communicated to you by the teaching staff after the creation of your group repository. This repository is used for java projects.
- "ids16gX-objectivec" for Objective-C code with Xcode tool, where X is the number of your group which will be communicated to you by the teaching staff after the creation of your group repository. This repository is only used for iPhone project.

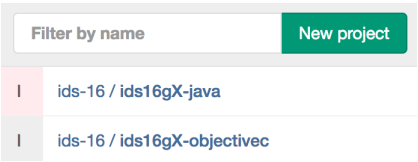


Figure 1: Repositories on gitlab (X is the number of your group)

## 8.3 Using Git with Netbeans and Xcode

In this section, you will discover how to use Git with Netbeans and Xcode. As it is described in Section 8.1, Git can be used in different modes. In this project, we will use the *centralized workflow* mode, which is very similar to the subversion (SVN) use[1]. In particular, we describe how to create a project, to upload it in your group Git repository on gitlab, to download it from a remote repository and to save your work in progress. ***In the following, you should use your gitlab username/password whenever there is a need for an authentication. Thus, see Section 5 if you do not know what are your gitlab username/password.***

### 8.3.1 NetBeans Built-in Git Client

This section describes the main steps of using Git with Netbeans. Firstly, a member of your group creates a project on her/his computer and upload it on your group repository for the Java code on gitlab. Then, the rest of group members download the project which is uploaded on you group repository for the Java code on their computers. During the project, you save and share your work using Git commands. Below, we describe each of these steps in more details.

#### A. Create a project and upload it in a remote Git repository

1. Create a new folder on your computer that will contain all your Java projects,
2. Create a new (or several) Java project(s) with Netbeans and save it (or them) in your folder created before,
3. Select your project(s) and right clic on them → "Versioning" → "Initialize Git repository" (***indicate the path of the folder that contains your Java project, not the path of your project(s)*** - see Figures 2 and 3),
4. Select your project(s) and right clic on them → "Git" → "Commit..." (see Figure 4),

[1]subversion (SVN) is another version control system older than Git.

5. Select your project(s) and right clic on them → "Git" → "Remote" → "Push...". If it is your first push, you should fill some fields, see Figure 5. Use the same url given in the figure (it is the url of your group repository for the Java code with ".git" at the end), just replace x with your group number. For username/password use the ones of your gitlab account.

**B. Download a remote Git project on your computer** Close all your Java projects opened and clic on "Team" (finder menu - see Figure 6) → "Git" → "Clone..." . Then, you should fill some fields, see Figure 7. Use the same url given in the figure (it is the url of your group repository for the Java code with ".git" at the end), just replace x with your group number. For username/password use the ones of your gitlab account.

**C. Save and share your work** For this project, you will often use these functionalities:

- Commit
- Push
- Pull

*Commit* is used to save *locally* (on your computer) the changes you have made to your projects. Hence, you will need to commit your projects every time you finish important step on them. To commit your projects, right-click them and select "Git" → "Commit...". In order to avoid conflicts between the versions of other members of your group and yours you should exclude XML files from the commits you make since such files are user-specific. You can also decide to commit only a specific file (right-click it and select "Git" → "Commit...").

*Push* is used to save *remotely* the changes you have made to your projects onto the Git server (your repository on gitlab). Usually, you will perform *commit* operations to save your work locally and after you will apply a *push* operation. Hence, you will need to push your projects every time you finish working on them. To push your projects, right-click them and select "Git" → "Remote" → "Push..."[2].

*Pull* is used to get the latest version of your projects. Because members of your team may have modified your projects while you were not working on them, you will need to make sure that you update your projects every time you start working. To update your projects, right-click them and select "Git" → "Remote" → "Pull...".

[2]Note that, there exists also a *Fetch* command which has a similar functionality as the *Push* command. However, contrary to a push, a fetch does not merges the branches.

For more information on how to use NetBeans' Git client, please follow the Netbeans official website: Using Git Support in NetBeans IDE [4]. If you want to use another way of use, you must follow the indication on [3] and the list of Git commands supported by Netbeans [4].

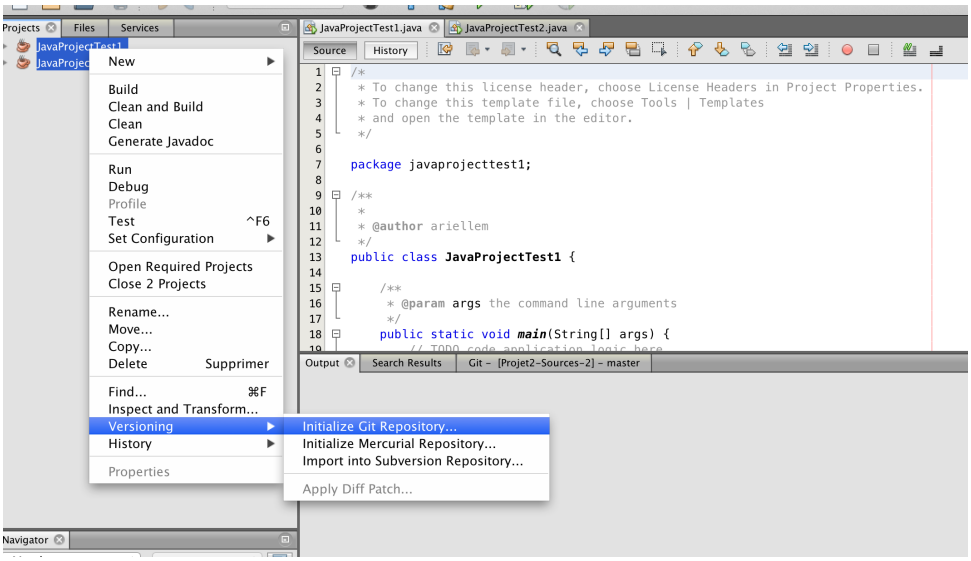


Figure 2: Initialize Git repository - Step 1

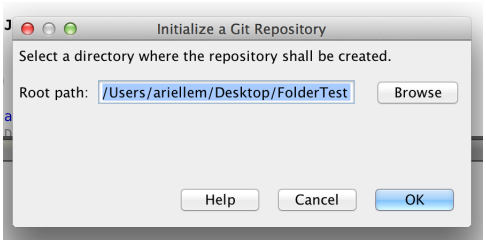


Figure 3: Initialize Git repository - Step 2

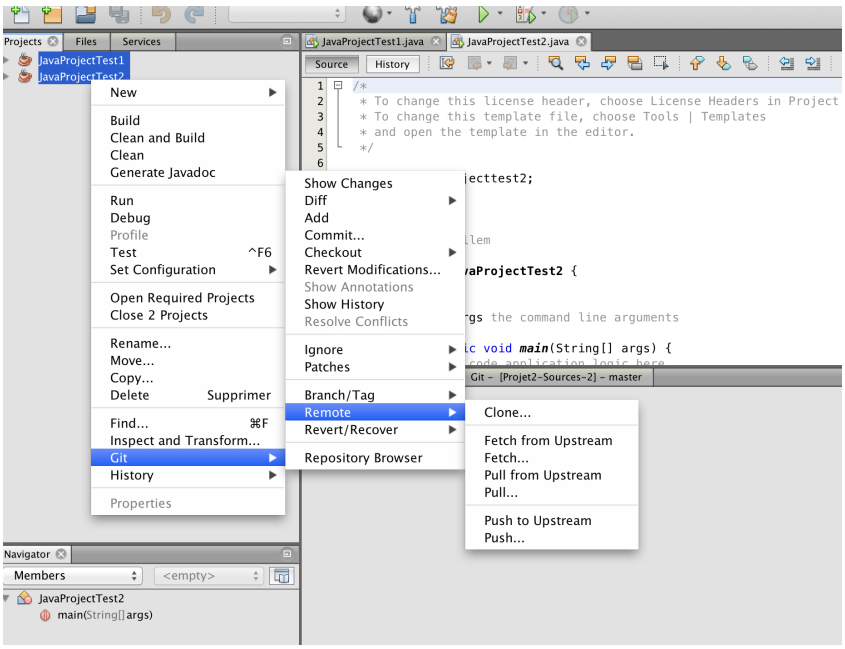


Figure 4: Git commands with Netbeans

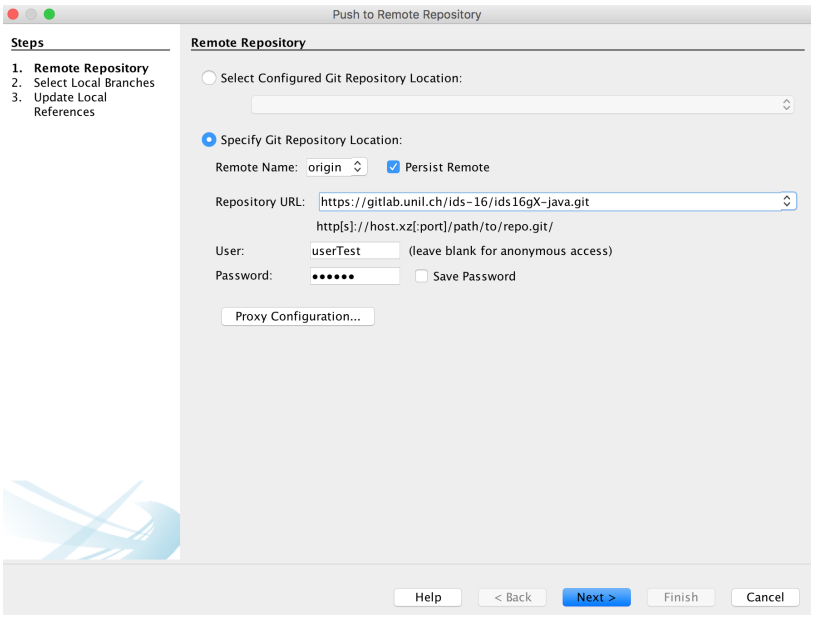


Figure 5: After your first Push - url of your group Java repository on gitlab with ".git" at the end (replace X by the number of your group), use your gitlab account's username/password in the corresponding fields

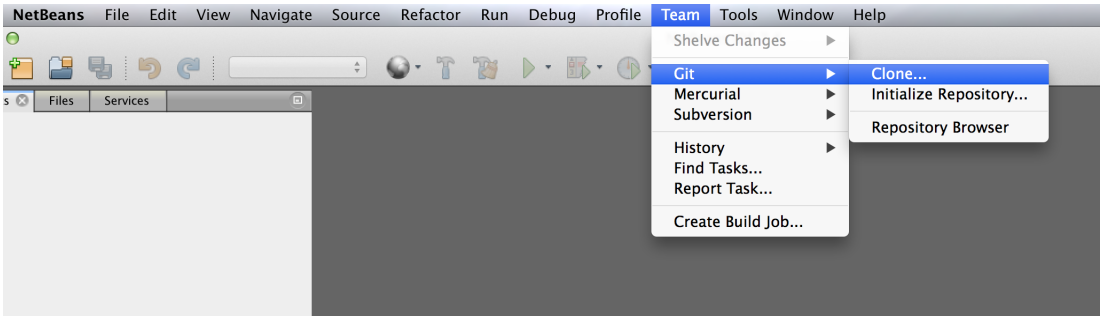


Figure 6: Clone Repository - Step 1

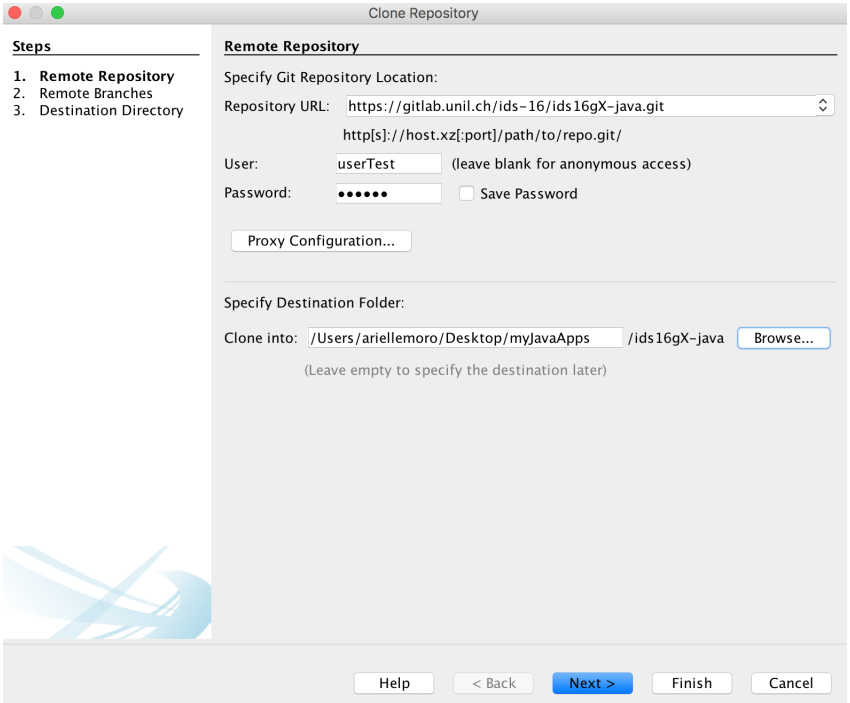


Figure 7: Clone Repository - Step 2, X is the number of your group

### 8.3.2 Xcode Built-in Git Client

This section describes the main steps of using Git with Xcode. Note that we use Xcode only for developing the iPhone application. Firstly, a member of your group creates a project on her/his computer and upload it on your group repository for the objective-C code on gitlab. Then, the rest of group members download the project which is uploaded on you group repository for the objective-C code on their computers. During the project, you save and share your work using Git commands. Below, we describe each of these steps in more details.

**A. Create a project and upload it in a remote Git repository**

- Create a new project on your computer with Xcode. When the storage location of the project is asked, choose "Create Git repository on "My Mac" (see Figure 8),
- Click on "Source Control" (finder menu) → select your working copie → Configure XXX ... (XXX = name of your project - see Figure 9),
- Click on "Remotes" → add a new remote repository by clicking on "+" button (See Figure 10). Then you need to fill some fields. See Figure 11. Use the same url given in Figure 11 (it is the url of your group repository for the objectivec code with ".git" at the end), just replace x with your group number and click on "Add Remote".
- Click on "Source Control" (finder menu) → "Push..." (see Figure 12).

**B. Download a remote Git project on your computer** Click on "Source Control" (finder menu) → "Check Out...". Then, you should fill some fields, see Figure 13. Use the same url given in the figure (it is the url of your group repository for the objectivec code with ".git" at the end), just replace x with your group number. If username/password required, you should use the ones of your gitlab account. After this step, Xcode will automatically download and open the project.

**C. Save and share your work** You will use the same functionalities as with Netbeans:

- Commit
- Push
- Pull

For each of them, click on "Source Control" (finder menu) and select your Git command.
For more information regarding Xcode built-in Git client, check `https://www.atlassian.com/git/workflows`.

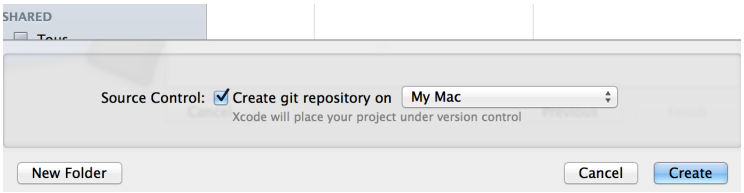


Figure 8: Create Git project with Xcode

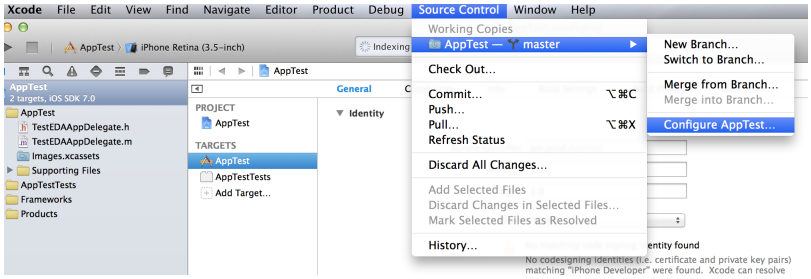


Figure 9: Add a remote repository - Step 1

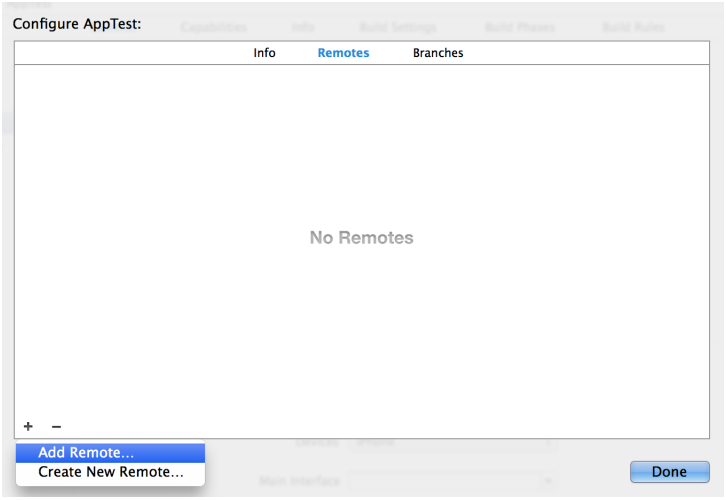


Figure 10: Add a remote repository - Step 2

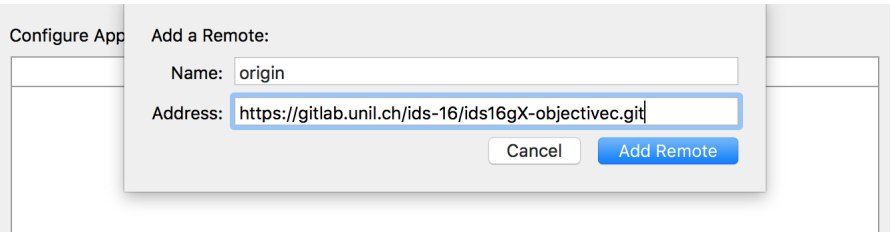


Figure 11: Add a remote repository - Step 3, X is the number of your group.

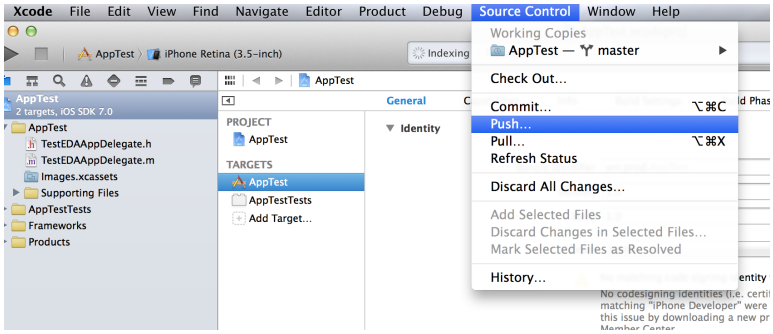


Figure 12: Git commands with Xcode

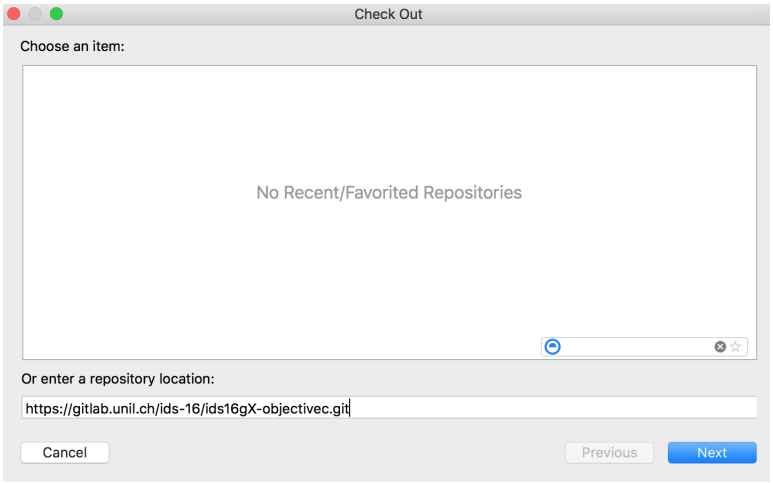


Figure 13: "Checkout" with Xcode. X is the number of your group.

### 8.3.3 Advice regarding the Management of Conflicts and file/project removal

Sometimes, if you perform a "Push", a conflict may appear because another member can have worked on it and performed a Push on it before you.

- If a conflict appears after a tentative of *push* operation, you will have to perform a *pull* operation. With this *pull* operation, you will be able to solve the conflict and commit it locally. Furthermore, it is very important to perform a *push* after this commit;
- In order to solve a conflict, it is better to compare the two versions and choose the parts that you want to save instead of merging;
- It is recommended to separate your work, i.e., at a given time it is preferable that only one student works on a file;
- If you want to delete a project or a file remotely, you must apply the changes locally and after, it will be applied remotely ( i.e., after *commit* and *push*). Sometimes, deletion doesn't work well with Netbeans or Xcode. In these cases, you can always use another Git client like SourceTree [6] (Clone repository → Delete files/projects → Commit it → Push it).

## References

[1] IDS Calendar.
*http://doplab.unil.ch/ids/.*

[2] Git tutorials.
*https://www.atlassian.com/git/tutorial.*

[3] Git workflows.
*https://www.atlassian.com/git/workflows.*

[4] Netbeans Guided Tour of Git.
*https://netbeans.org/kb/docs/ide/git.html.*

[5] Xcode + Git tutorial.
*http://www.appcoda.com/git-source-control-in-xcode/.*

[6] SourceTree application.
*http://www.sourcetreeapp.com.*