Project Guidelines

This document provides all information regarding the project rules, organization and deadlines. Hence, it is ***very important*** to read it carefully in order to know the rules and also to ensure a good progress of your project.

Contents

1 Project Overview

The complete description of the project can be found in the *Project overview* document available on our website (see: `http://doplab.unil.ch/eda-project/`).

2 Software/Hardware Requirements

For each phase, we will use the project tools presented at the beginning of the semester that you can find at this link: `http://doplab.unil.ch/wp-content/uploads/2017/02/IntroductionLabProjectTools.pdf`. Additionally, we use *Git* as version control system. As Git clients, students can use *Netbeans built-in Git client* for Java code and *Xcode built-in Git client* for Swift code. Instead of using Netbeans and Xcode built-in Git clients, they can also decide to only use *SourceTree*. For more information about Git, please read Section 7 (Appendix).

3 Prerequisite

The prerequisite of this course is the course *Introduction to Distributed Systems (IDS)* (see: `http://doplab.unil.ch/ids/`). So, in order to perform the project, you need a good knowledge of Java distributed programming, average knowledge of Swift or a strong will to learn Swift and iPhone programming in a short time duration (about 2 or 3 weeks).

4 Organization

In order to do the project, students must form groups. Groups are formed at the beginning of the semester. Each group will create two Git repositories (one for the Java projects and another for the Swift project) on `https://gitlab.unil.ch/` where code and the documents of their project should be found at each phase. Please read carefully Section 7 (Appendix) to know more about the Git repositories and how you can use them for the development of you project.

For organizational reasons, such as ***creation of the Git repositories***, some steps should be done. These steps are described below.

4.1 Step 1 - Course registration

All students, regardless of their university, must follow this link `http://doplab.unil.ch/eda-registration/` and edit the required fields of the Google document indicated. The deadline for the course registration is Wednesday 1st March, 2017 at 23:59. EPFL students in particular, need

to perform some additional steps for the registration to this course. They should go to this link (`http://ic.epfl.ch/cycle_master_en`), and then follow the instructions under the item *"In case you take a course outside the study plan at UNIL-HEC, ..."*.

4.2 Step 2 - Creation of a Gitlab account

During the project, you will use your Gitlab username/password to connect to your project repository on `https://gitlab.unil.ch/` and to be able to create the two repositories needed. 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, you already have a Gitlab account. Your Gitlab username/password are the same as the one's used to connect to your UNIL mailbox at `https://owa.unil.ch/`.
- **Other students (including EPFL students):** if you do not have an email address at UNIL, you will create your Gitlab on this website: `https://gitlab.unil.ch/users/sign_in`. A confirmation email regarding the creation of the Gitlab account should be sent to your academic mailbox. The confirmation email contains your Gitlab username/password.

4.3 Step 3 - Group formation

Each group should be composed of 3 members. Then, each group must 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, in the window "Sign in" ("LDAP" for UNIL students and "Standard" option for other students - see Figure 1), login with Gitlab username and password (see Section 4.2 if you do not know what are your Gitlab username/password). If you do not have a Gitlab account yet, please follow the instructions in Section 4.2. In case you have any problem to login with your Gitlab account's username/password, please contact `Michel.Schuepbach@unil.ch`.
   Once logged in, you can change your password and then close the browser window. Note that ***it is very important that each group member does this process***. In fact, it enables you to create your group Git repositories on Gitlab. You will use your group Git repositories for the project.

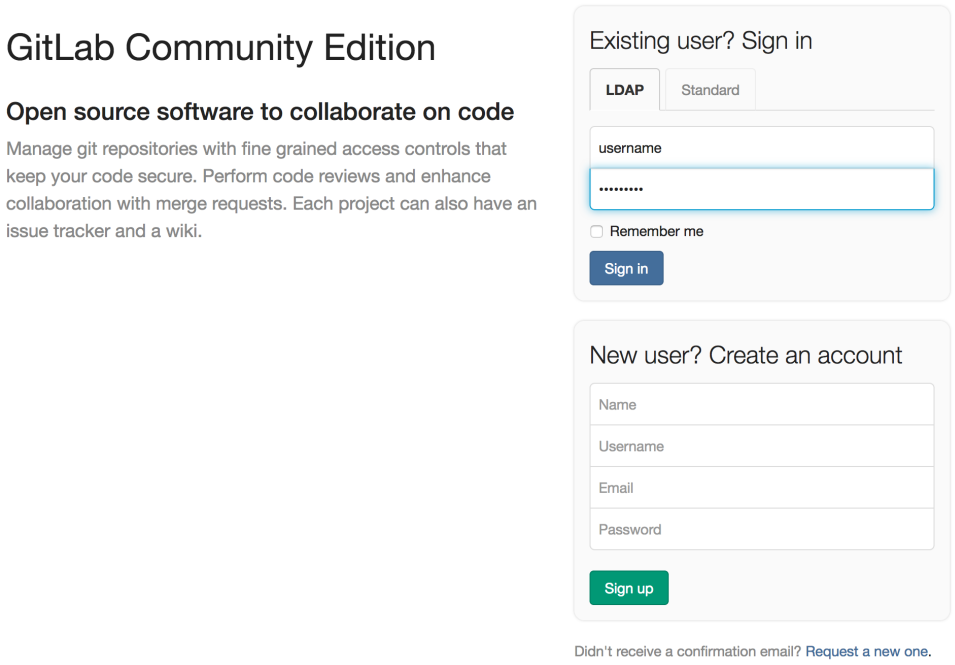

Figure 1: Account activation on Gitlab

2. Once the above (activation of an account on Gitlab) is done for all group members; each group should send to `arielle.moro@unil.ch` an email until **Monday, 6th March, 2017 at 23:59**, and announces its group formation. Of course, one email per group is sufficient. Your email should respect the following format:

   - **Email object:** eda17groupformation
   - **Email content:** the list of the group members as well as a name for your group. For each member you should indicate: (first name, last name, academic email address). If you are less than 3 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***.

4.4 Step 4 - Git repositories creation

Once Step 1 and Step 2 are done, each group will be able to create Git repositories (see Section 7.2 below).

5 Submission deadlines

At the end of each phase of the project, your project code (Java + Swift) and the PDF format of your presentations slides should be found on the Git repository of your group before the indicated deadline. These deadlines are:

- ***Phase 1:*** Wednesday, 22th March, 2017 at 23:59.
- ***Phase 2:*** Wednesday, 26th April, 2017 at 23:59.
- ***Phase 3:*** Wednesday, 31th May, 2017 at 23:59.

Note that we only check your Git repository on `https://gitlab.unil.ch/` and on no other personal repositories. Furthermore, ***you do not need to provide a report for the project*** but we advise you to comment your code.

6 Presentations

At the end of each phase of the project, you have to do a short presentation explaining the work you have done. To know the presentation dates, please check the calendar on our web site [1].

The exact time and place for each group presentation will be announced by email a few days before each presentation. The presentations will be closed and private (like an oral examination) and will take place every 15 minutes (the last 5 minutes are for our questions). The presentations can be in English or French. All group members should be present during the presentation of their group. 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). In the case where a member is absent, she/he should send us an email explaining the reason of the absence. If you take this course, we assume that you keep free your Thursday mornings 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. However, if you have a valid reason to be absent and if your group mates do not have a problem with your absence, you can inform us by sending an email and can be absent during the presentation. In this case, you will be graded as the others.

Each presentation will comprise three parts:

- First part (about 5 min.): a demo of your application. The demo can be done on your laptops or one of the Mac (Internef 143) computers. However, if you are an EPFL student we advise you to use your laptops for demos because your access to room Internef 143 is only guaranteed

during the exercise sessions. Note that for the demos, all elements of your application (e.g., application server, rich client, iPhone client, etc...) can run on the same machine (localhost).

- Second part (about 5 min.): presenting the slides. All the following slides should present:
  - Architecture of your application: main building blocks of your application (e.g., web tier, business tier, iPhone client, etc...);
  - Communication between building blocks (e.g., HTTP, JMS, etc...);
  - Work distribution among group members.
- Third part (about 5 minutes): Questions & Answers.

Some advice for your presentations: (1) Do not repeat the demo in the slides by using screen shots and (2) get prepared (so we do not lose time).

7 Appendix

7.1 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/tutorials/comparing-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.

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 Swift program).

7.2 Gitlab web interface and Git repository creation

For this project, one member of the group can now create the two Git repositories on `https://gitlab.unil.ch/`. Once logged in with your with Gitlab username and password (see Section 4.2 if you do not know what are your Gitlab username/password), you can see your dashboard and create the two repositories by clicking on **+New Project** as indicated in Figure 2.

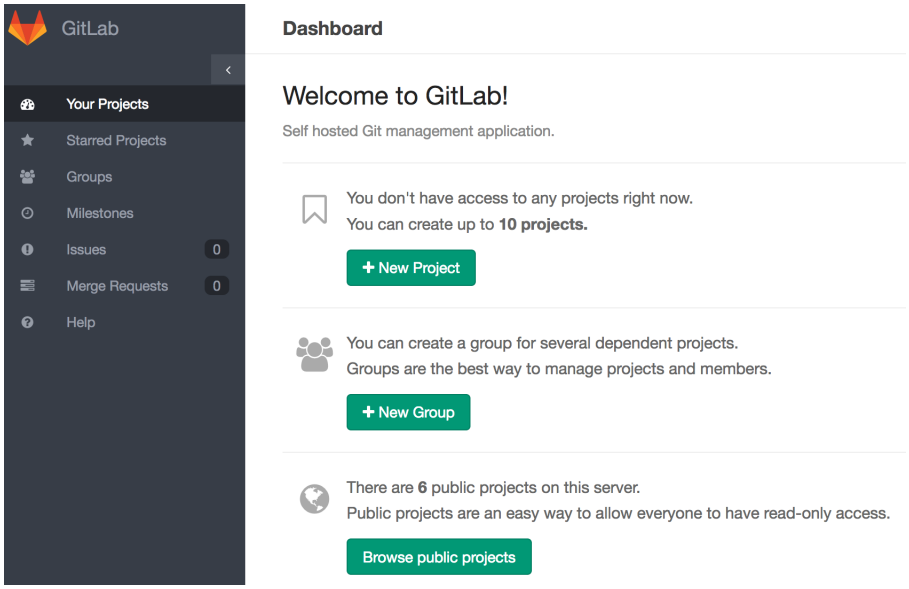

Figure 2: Gitlab dashboard

Thus, for this course you will have to create the two private following repositories:

- "eda17-*groupname*-java" for Java code with Netbeans tool, where *groupname* is the name of your group that you previously chose during the group formation step. This repository will be used for Java projects.
- "eda17-*groupname*-swift" for Swift code with Xcode tool. This repository will be used for Swift projects.

After their creation, you will be able to see the two projects on your Gitlab dashboard and you will also be able to easily add all the remaining group members to the project by clicking on it and on **Members** on the left menu of the projet. If and only if the all the group members already have or have created their Gitlab account as previously indicated, they can be added to the projects. You also need to add the professor and the teaching staff to these two Git repositories as members of the project because we need an access to check your work at the end of each phase.

7.3 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 7.1, 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 4.2 if you do not know what are your Gitlab username/password.***

7.3.1 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 3 and 4),

4. Select your project(s) and right clic on them → "Git" → "Commit..." (see Figure 5),

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 6. 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, you must use the ones of your Gitlab account.

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

B. Download a remote Git project on your computer Close all your Java projects opened and clic on "Team" (finder menu - see Figure 7) → "Git" → "Clone..." . Then, you should fill some fields, see Figure 8. 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...".

For more information on how to use NetBeans' Git client, please follow the Netbeans official website: Using Git Support in NetBeans IDE [3].

[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.

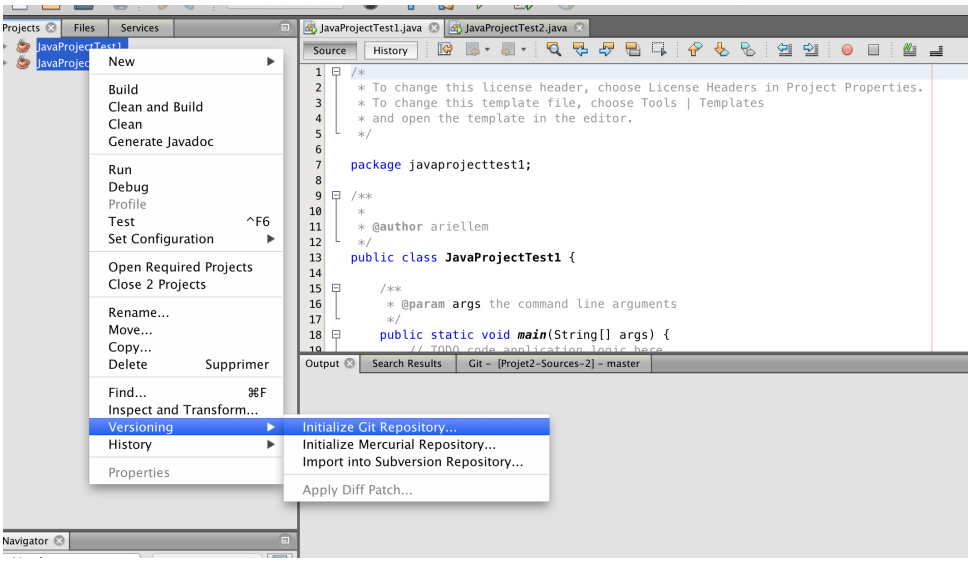

Figure 3: Initialize Git repository - Step 1

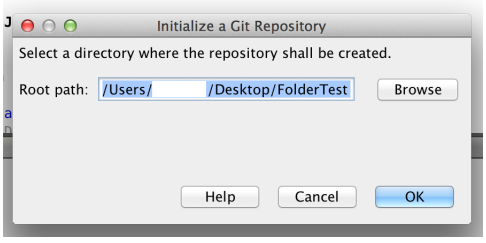

Figure 4: Initialize Git repository - Step 2

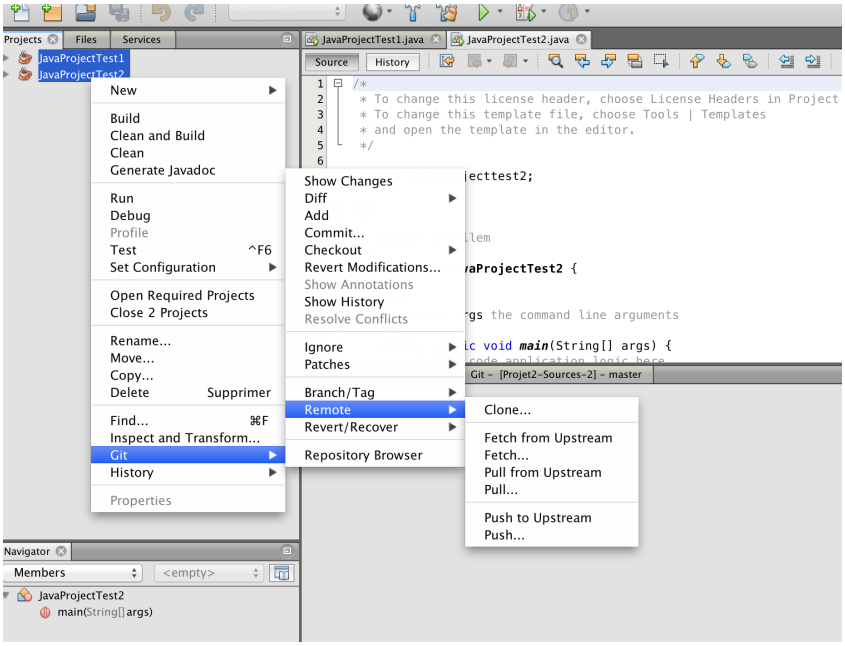

Figure 5: Git commands with Netbeans

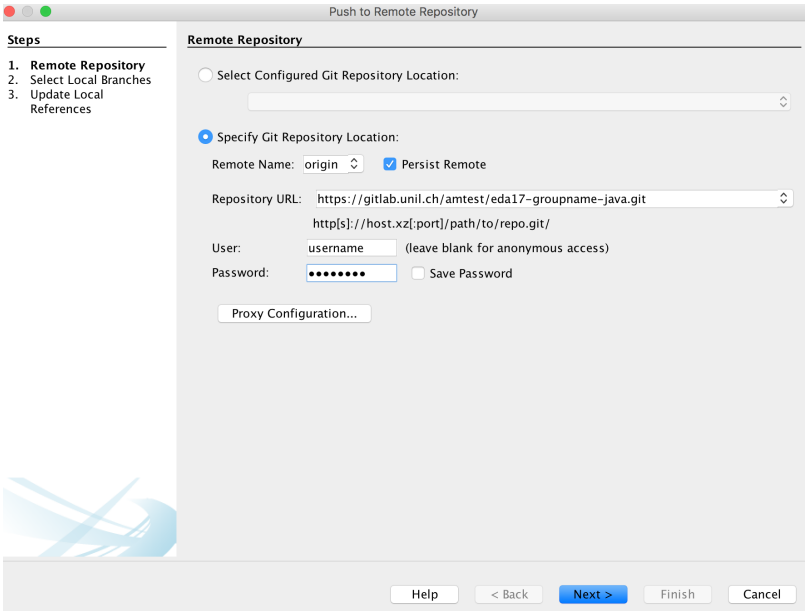

Figure 6: After your first Push - url of your group Java repository on Gitlab with ".Git" at the end (where *groupname* is the name of your group that you previously chose during the group formation step), use your Gitlab account's username/password in the corresponding fields

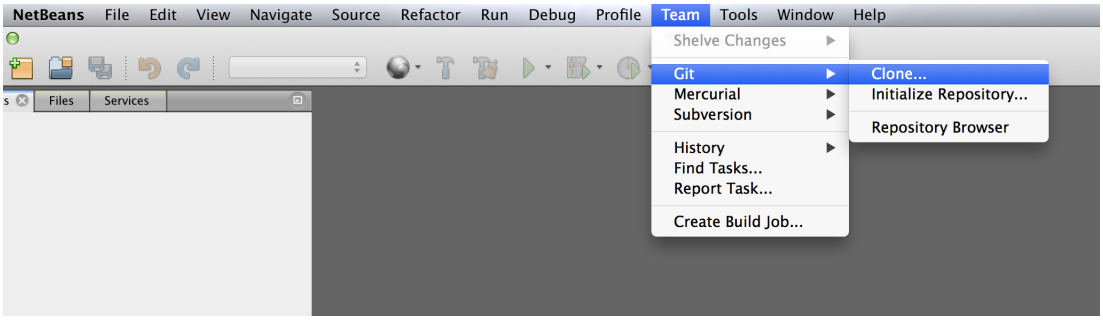

Figure 7: Clone Repository - Step 1

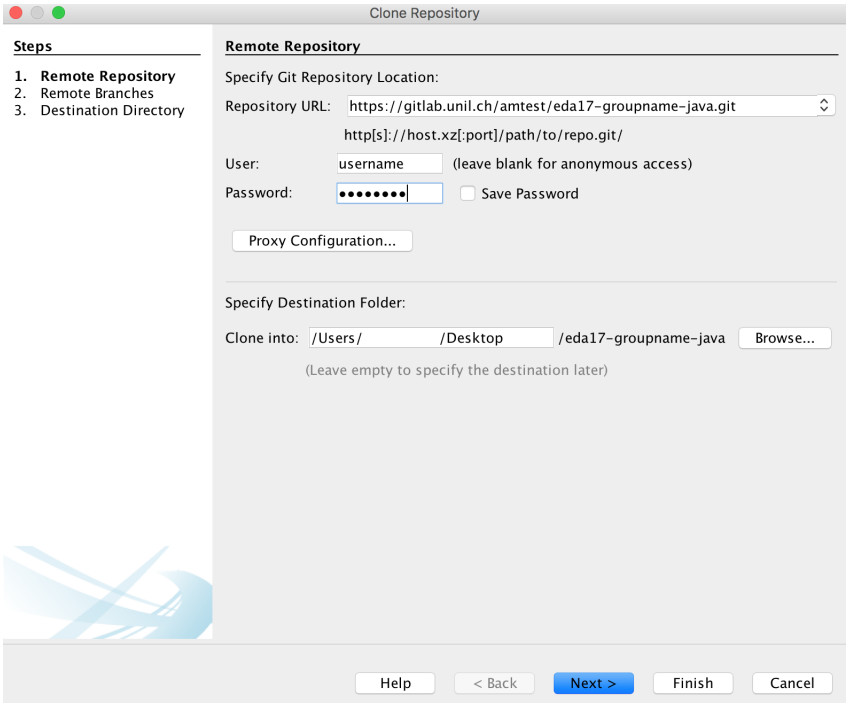

Figure 8: Clone Repository - Step 2, where *groupname* is the name of your group that you previously chose during the group formation step

7.3.2 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 XCode project on her/his computer and upload it on your group repository for the Swift code on Gitlab. Then, the rest of group members download the project that is uploaded on you group repository for the Swift 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 9),
- Click on "Source Control" (finder menu) → select your working copie → Configure XXX ... (XXX = name of your project - see Figure 10),
- Click on "Remotes" → add a new remote repository by clicking on "+" button (See Figure 11). Then you need to fill some fields. See Figure 12. Use the same url given in Figure 12 (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 13).

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 14. 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.

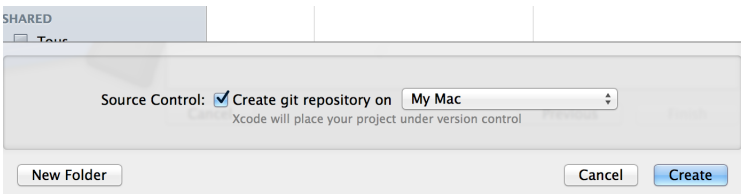

Figure 9: Create Git project with Xcode

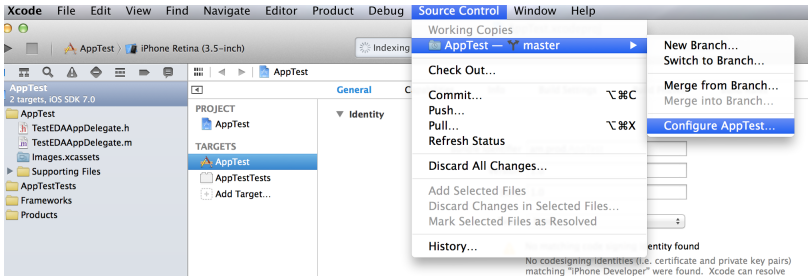

Figure 10: Add a remote repository - Step 1

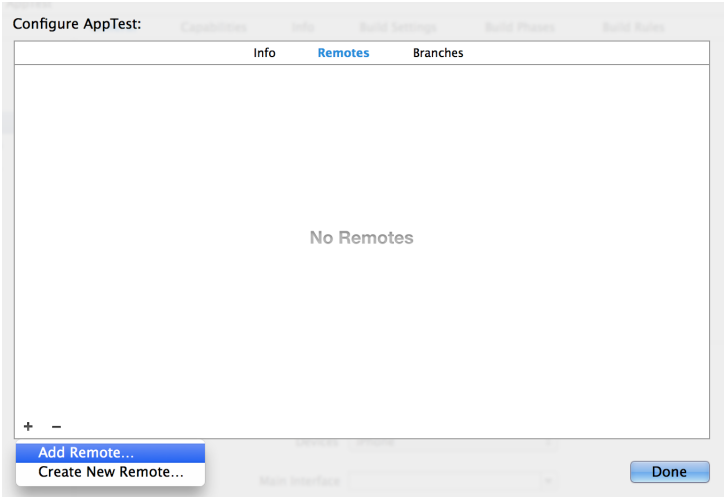

Figure 11: Add a remote repository - Step 2

7.3.3 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

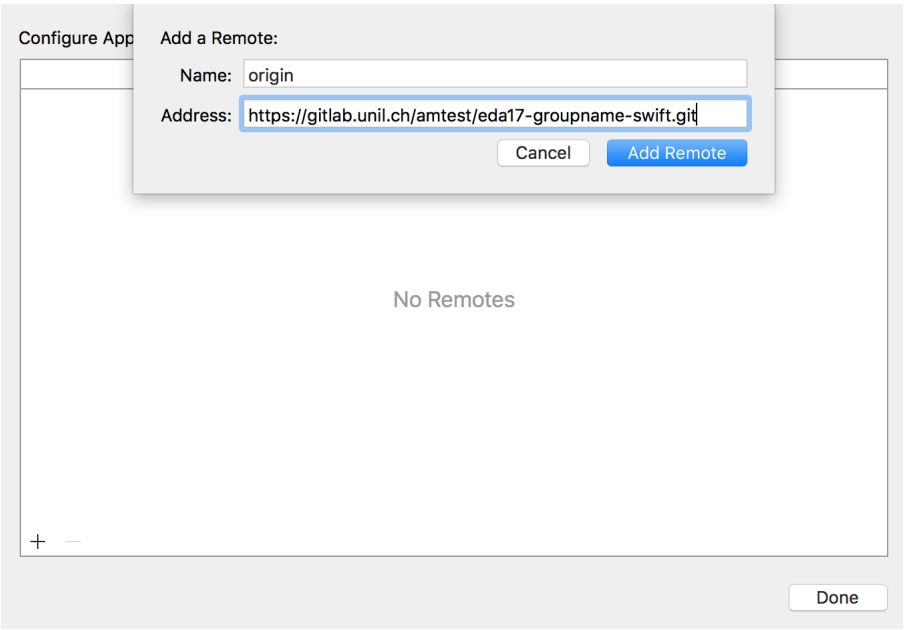

Figure 12: Add a remote repository - Step 3, where *groupname* is the name of your group that you previously chose during the group formation step.

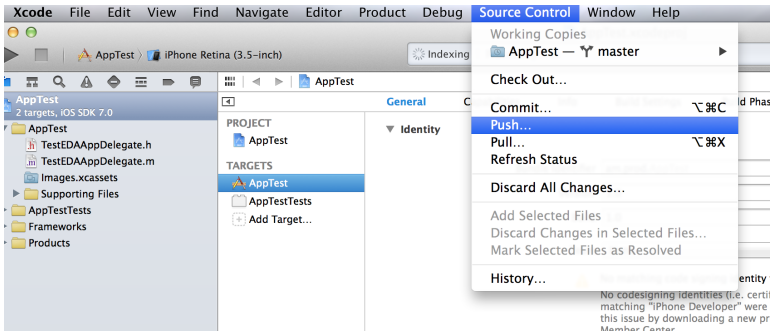

Figure 13: Git commands with Xcode

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

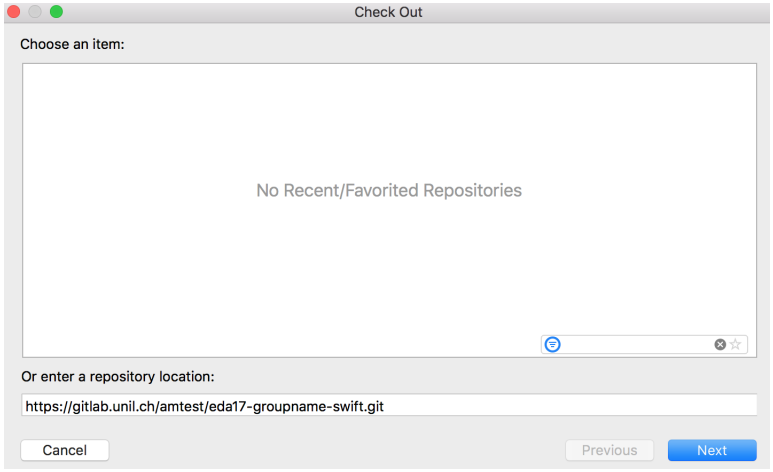

Figure 14: "Checkout" with Xcode, where *groupname* is the name of your group that you previously chose during the group formation step.

Xcode. In these cases, you can always use another Git client like SourceTree [5] (Clone repository → Delete files/projects → Commit it → Push it).

References

[1] EDA Calendar.
http://doplab.unil.ch/eda/.

[2] Git workflows.
https://www.atlassian.com/git/tutorials/comparing-workflows.

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

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

[5] SourceTree application.
http://www.sourcetreeapp.com.