Git

В рамките на курса по **FrontEnd Програмиране**, всички задачи от упражнения, домашни и проекти ще се предават и проверяват чрез програма за разпределена разработка и контрол на версиите, наречена git.

Принцип на работа

Програмата Git, работи на следния принцип:

Имаме *проект* (папка с работни файлове), по който работят няколко човека едновременно. За да осигурим постоянна синхронизация между работата на всеки от екипа, поставяме проекта под *version control*.

Това означава, че чрез програмата **git**:

1. задаваме проследяване на промените по проекта
2. поставяме копие на този проект на **git** сървър

Задаване на проследяване на промените по проекта

Когато имаме проект, който искаме да поставим под *version control*, изпълняваме командата **git init** в основната папка на проекта, с което инициализираме **локално git repository** за този проект.

Git repository се нарича всяка папка на проект, която съдържа в себе си скрита папка: **.git**. В тази папка програмата **Git** пази информация за направените промени по файловете в проекта.

Т.е. това което прави програмата **Git**, е че създава една скрита папка в папката на проекта ни и там си записва какви промени са били извършени по файловете, кога и от кого.

Именно по този начин чрез други инструменти на програмата **Git**, като *checkout*, *commit и т.н.*, можем да управляваме различните версии на кода в проекта.

Поставяне на копие на проекта на **git** сървър

Когато сме създали *локално git repository*, ние можем да управляваме версиите на проекта, *единствено* на същия компютър на който се намира *repository*-то.

За да могат всички останали от екипа да синхронизират работата си с това *repository*, е необходимо, то да се намира на *git сървър*. Поставянето на едно *repository* на *git сървър*, го прави *основно* и затова го наричаме **origin**.

Всеки, който иска да работи по проекта може:

1. Да го клонира и да изпраща промените си директно към **origin**-а (може да стане само ако е вписан като collaborator в настройката на repository-то)
2. Да го **fork**-не и да изпраща промените си към свое копие на **origin**-а, след което да качва промените си чрез заявка за **merge**, наречена **Pull request**

Инсталация на GitHub Desktop и настройка на работния проект

Какво е GitHub

В рамките на курса, ще използваме платформата **GitHub** за **git server** на *курсовото repository*: **swift-academy-homeworks**.

GitHub е уеб платформа, която ни предоставя услугата **git hosting**. Това означава, че **GitHub** дава възможност да се складират **git repositories** на техен сървър.

Какво е GitHub Desktop

GitHub Desktop е десктоп прогрма, която е специализирана за работа с **GitHub** проекти. Тази програма представлява GUI (*Graphic User Interface*) на програмата **Git**.

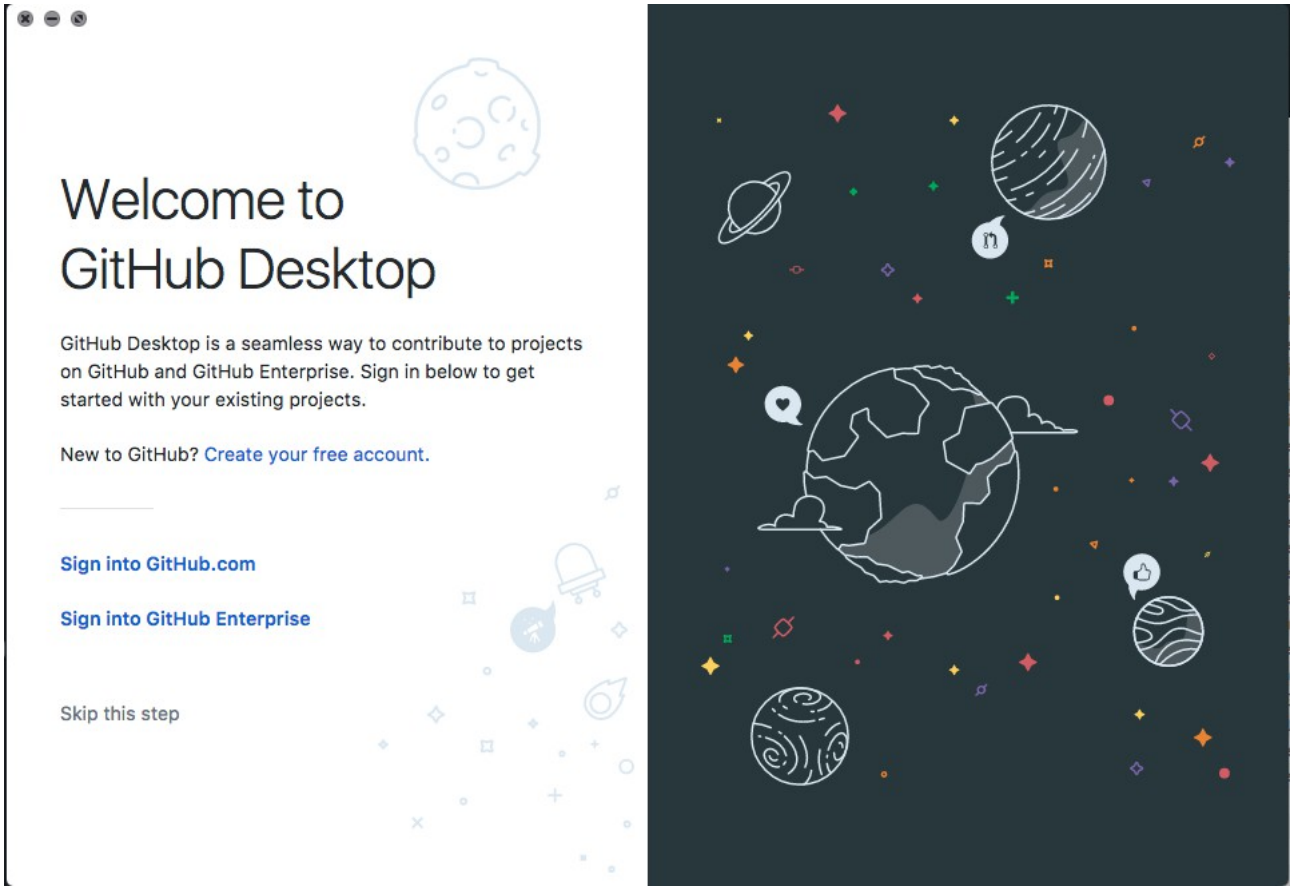
Чрез **GitHub Desktop**, можем да изпълняваме всички *git команди*, посредством графичен интерфейс (т.е. менюта и бутони), вместо в *git конзолата* (стандартната текстова среда на git).

Инсталация на GitHub Desktop

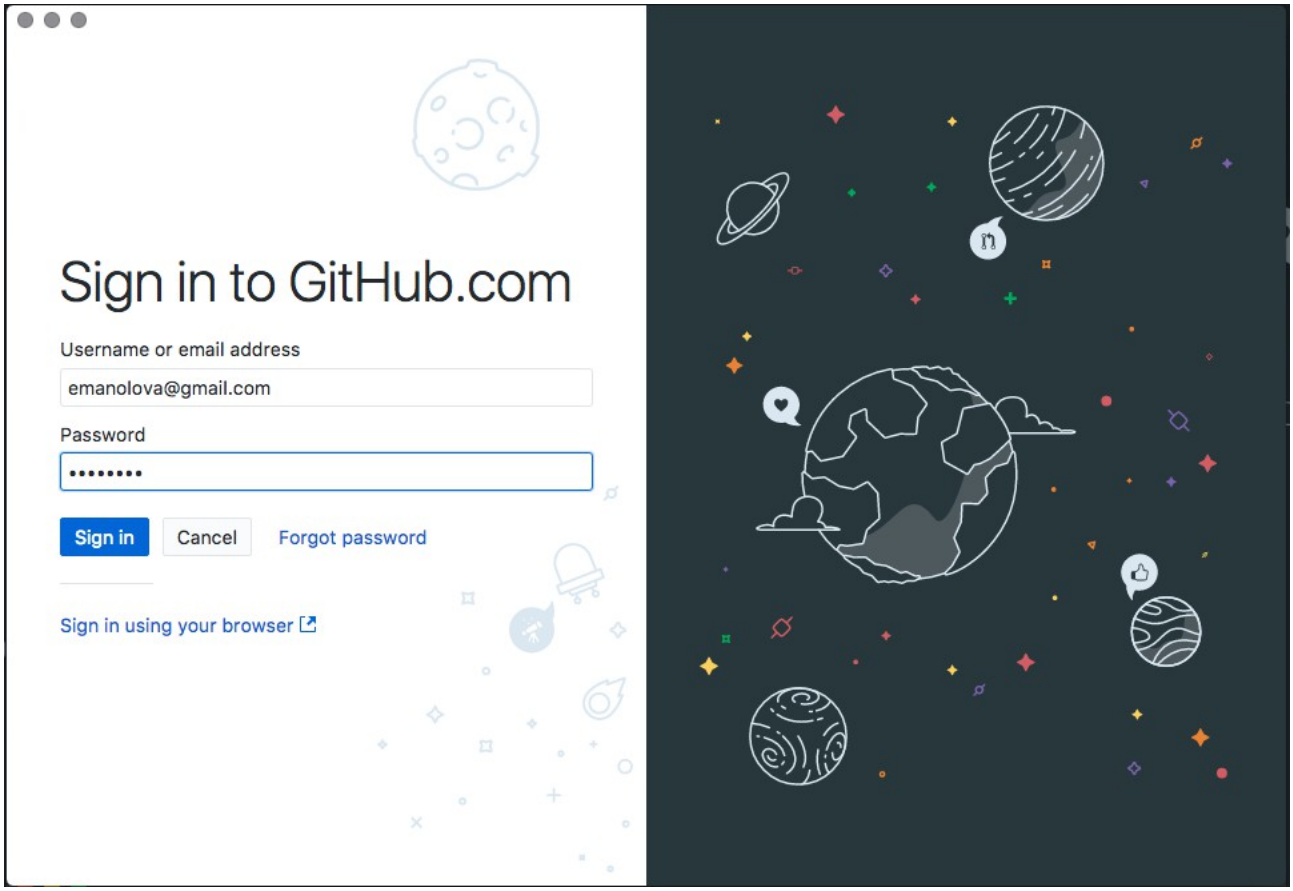
Програмата може да се download-не директно от сайта на GitHub*: https://desktop.github.com/

* Официлано се поддържат дистрибуции за Windows и MacOS операционни системи. За линукс системите евентуално може да се намерят build-ове ето тук: https://github.com/gengjiawen/desktop/releases

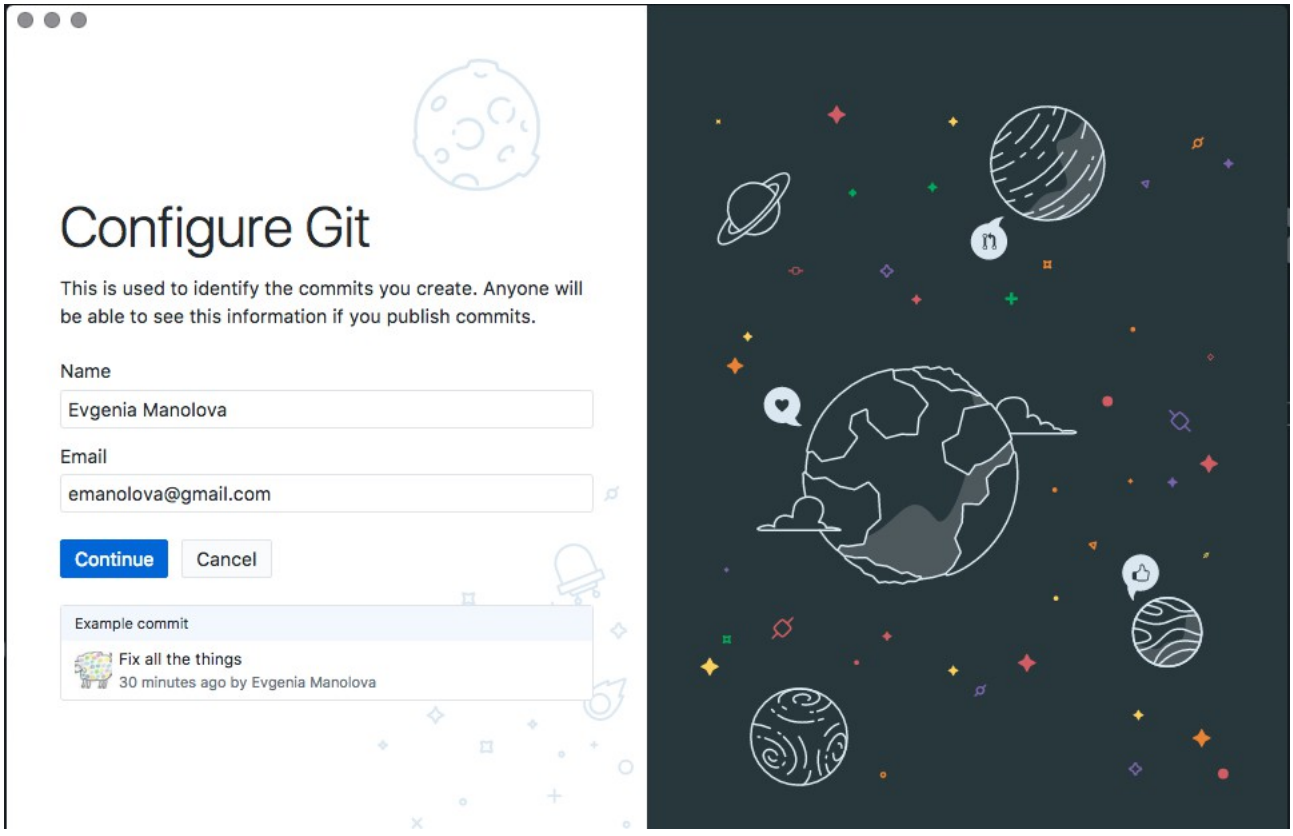
След като инсталирате **GitHub Desktop** ще се зареди прозорец за логин в **GitHub**:



Ако все още нямате акаунт в **GitHub**, сега е момента да си направите такъв. Когато сте готови, преминете към логин:



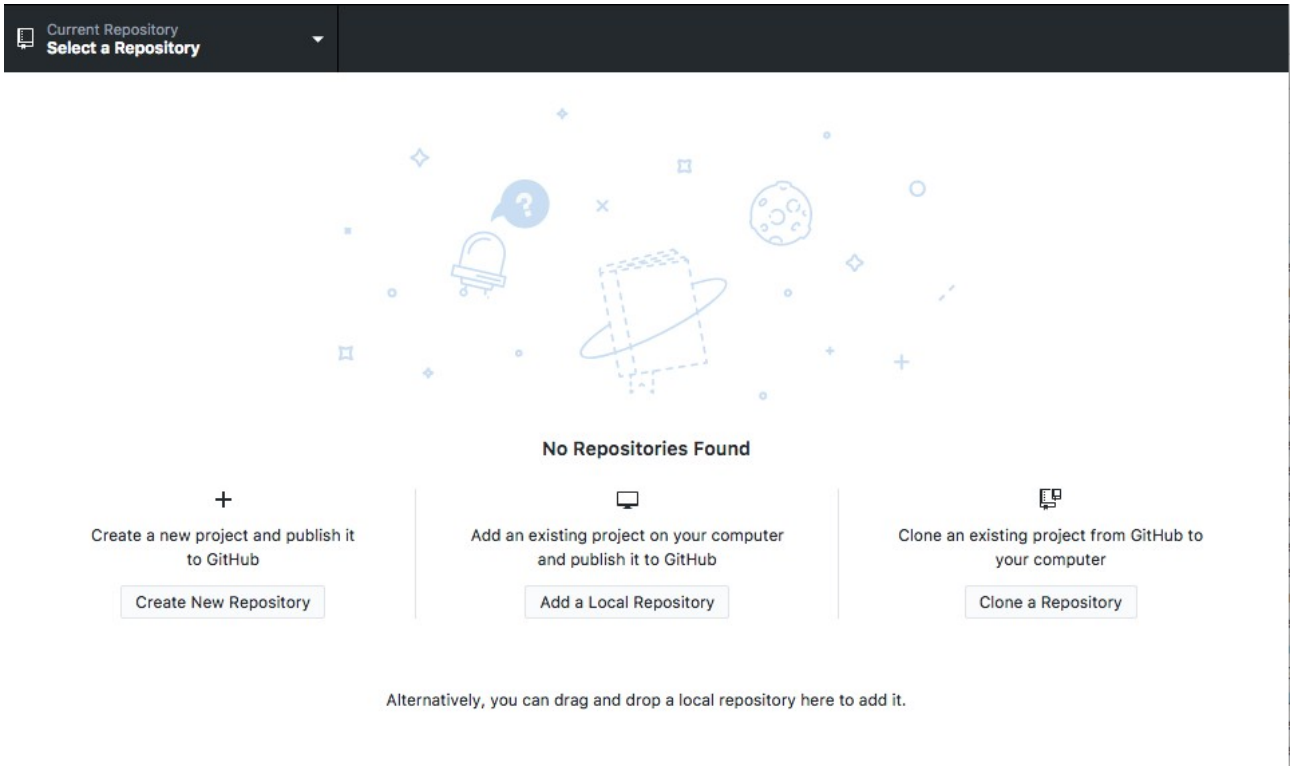
При успешна аутентикация, ще видите следния екран, в който *задължително* трябва да въведете валидни данни (две имена и имейл), иначе програмата няма да работи:



Настройка на работния проект

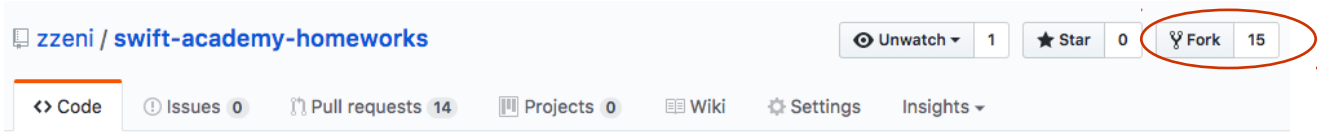
Fork

Ако сте инсталирали и инициализирали успешно програмата GitHub Desktop, би трябвало при стартирането ѝ да виждате ето този екран:



Време е да създадете своя работен проект.

Това става, като отворите следния линк: https://github.com/zzeni/swift-academy-homeworks и кликнете върху бутона **Fork** (трябва да сте логнати в GitHub сайта):



Това, което ще се случи е, че проекта **swift-academy-homeworks**, ще се копира във вашия профил. Т.е. вие ще си създадете *собствено repository*, което е копие на zzeni/**swift-academy-homeworks** и което приема за свой ***origin*** отново zzeni/**swift-academy-homeworks**.

Когато работите по задачите от упражнения, домашни и проекти, вие ще създавате и контролирате файловете в собственото си *repository*, но за да ги предадете, ще трябва да правите ***Pull request***.

Pull request-а е просто заявка за прехвърляне на вашите промени, от вашето репозитори в ***origin***-а.

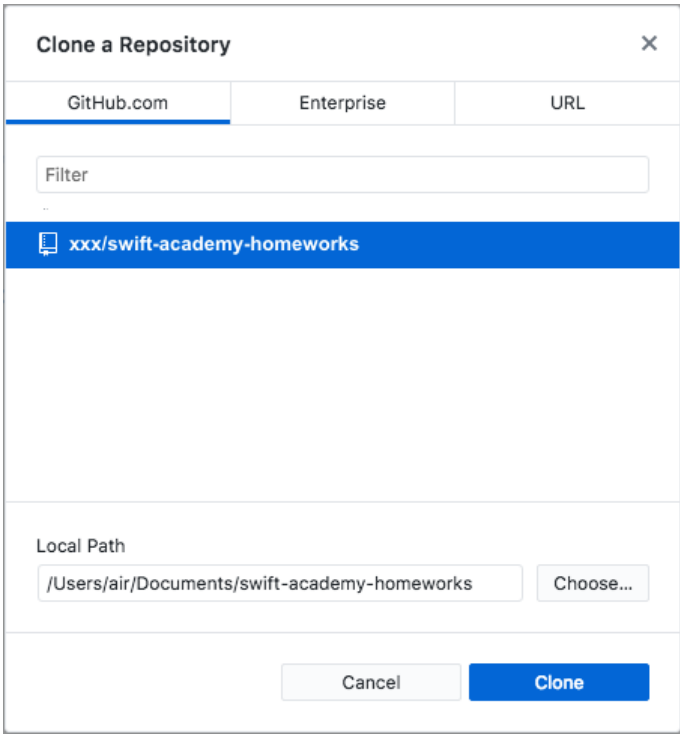
По този начин, аз ще мога да преглеждам кода ви и съответно да го ***merge***-вам в базовия проект (*origin*-а).

Clone

Когато вече сте **Fork**-нали базовото repository, вече можете да го *изтеглите* на вашия компютър с помощта на **GitHub Desktop**.

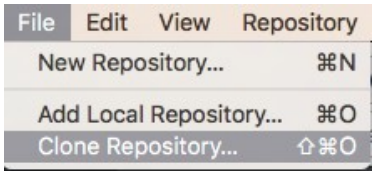
Натиснете бутона **Clone Repository** и изберете от списъка **swift-academy-homeworks**.

Преди да натиснете бутона **Clone**, изберете мястото, където да се създаде курсовият проект. Подходящи локации са: **My Documents**, **Desktop** или **D:\Projects**.



Важно: необходимо е да имате write права за папката, в която изберете да се клонира проекта!

Друг начин да клонирате repository, е чрез менюто: **File -> Clone Repository**:



Забележка:

Всяко ***git repository*** съдържа освен работните фйлове от проекта и една папка **.git**, която е скрита. Много важно е да не пипате тази папка, както и всички други файлове, чиито имена започват с **.git**

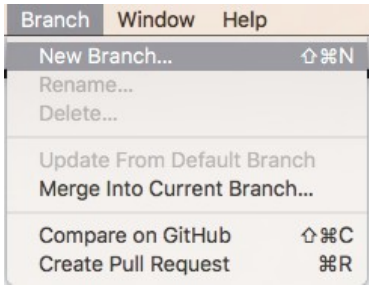
Branch

Преди да започнете работа в ново-клонирания проект, първо трбва да създадете свой **Branch**.

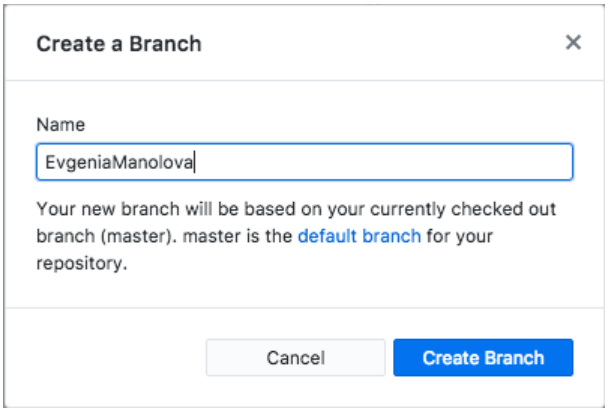
Branch-овете са нещо много удобно, което ни помага да не си пречим с останалите от екипа докато работим. Конкретно в нашият случай - *собственият branch*, ще ви осигури едно лично пространство, в което да работите само по вашите задачи, без да получавате решенията на другите курсисти.

Малко повече за *git branches*, ето в това видео: https://www.youtube.com/watch?v=w3jLJU7DT5E

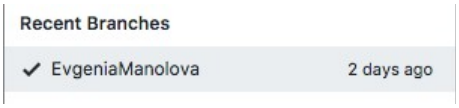
За да си направите *branch*, достатъчно е просто да изберете **Branch -> New Branch** в програмата **GitHub Desktop**:



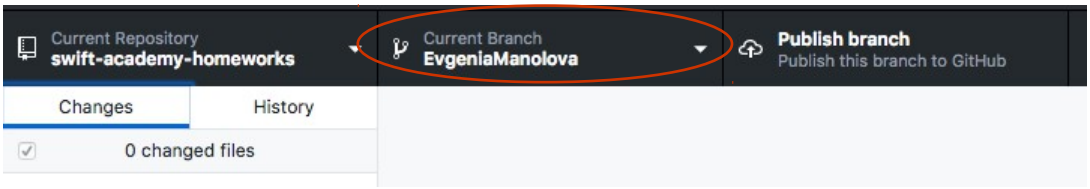
Именувайте *branch*-а, като изпишете двете си имена на латиница и *без интервал* между тях:



При създаването, ще видите *prompt* меню, от което можете да изберете новосъздадения branch за *работен branch*:



За да се уверите, че успешно сте създали вашият *branch*, сверете дали името е изписано под **Current Branch** от навигационната лента:

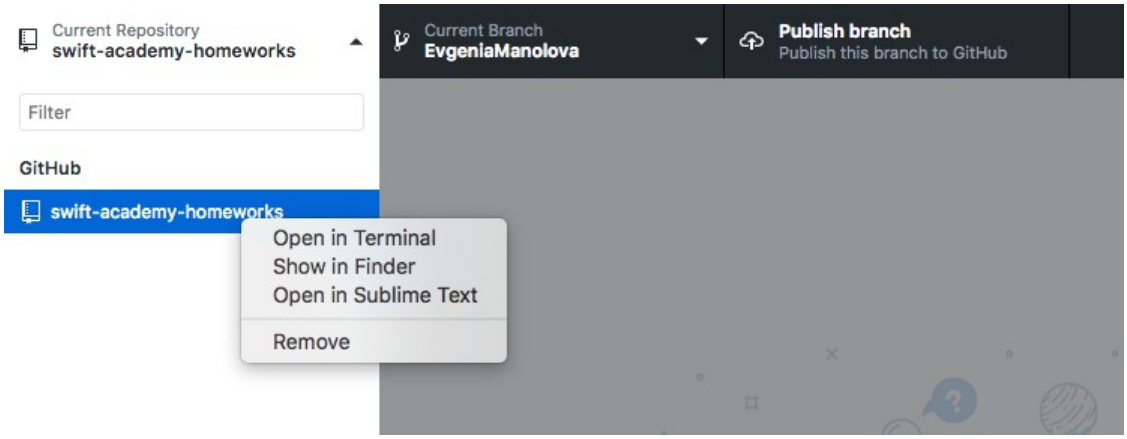


Сега вече сте готови за работа!

Интеграция с Brackets

Има много начини за достъпване на папката на проекта през **GitHub Desktop**.

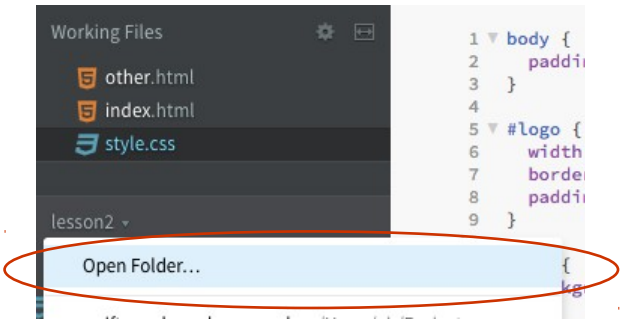
Един от тях е, чрез кликване на **Current Repository**, след което клик с десен клавиш на мишката и избор от падащото меню **Show in Explorer** (за MacOS, текста е *Show in Finder*)



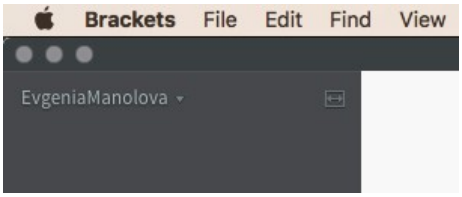
Друг начин е през менюто **Repository**:

След като отворите папката на проекта в **Explorer**, трябва да създадете в нея една нова папка, именувана с вашето име *на латиница*.

Стартирайте **Brackets** и отворете новосъздадената папка през менюто **File -> Open Folder** или с десен клик върху името на текущия проект и избор на **Open Folder** от менюто:



Намерете **swift-academy-homeworks** проекта и изберете вашата папка, след което кликнете **Open**. Трябва да видите нещо подобно в **Brackets**:



Правила за именуване и работа

След като сте задали своята папка за работен проект в **Brackets**, остава само да започнете да решавате задачите от курса в този проект.

За да не се обърквате и за да ме улесните в проверяването, следвайте следния шаблон за именуването на вашите папки и файлове:

- Всички задачи към даден урок се поставят в папка, именувана спрямо номера на съответния урок: **lesson1, lesson2** и т.н.
- Всяка отделна задача за упражнение, се решава в отделен подпроект (папка), именуван: **task1**, **task2** и т.н. , а всяка такава папка се поставя в папката на съответния урок
- Всяка отделна задача за домашно, се решава в отделен подпроект (папка), именуван: **hw1**, **hw2** и т.н. , а всяка такава папка се поставя в папката на съответния урок (**lesson1, lesson2)**
- Папките на задачите от контролните се именуват: **test1**, **test2** и т.н.
- Папката на финалния проект се именува: **project**.

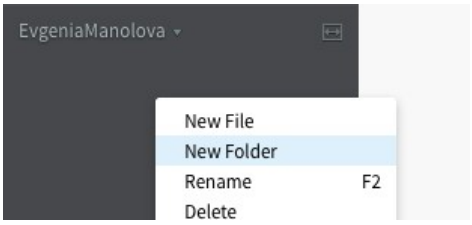
ВАЖНО: Работи се единствено в собствената папка!

Пример:

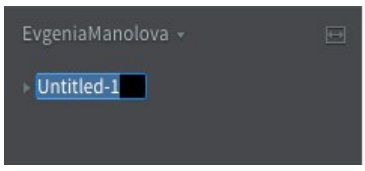
- Когато работя по дадена *задача за упражнение*, напр. *Задача 1* от *урок 3*, променям файловете в папката **EvgeniaManolova -> lesson3 -> task1**.
- Когато работя по дадена *задача за домашно*, напр. *Задача 2* от *домашно 3*, променям файловете в папката **EvgeniaManolova -> lesson3 -> hw2**.
- Когато работя по дадена *задача от контролно*, напр. *Задача 1* от *контролно 2*, променям файловете в папката **EvgeniaManolova -> test2 -> task1**.

За да започнете да работите по конкретна задача, можете да си създадете необходимите папки директно от **Brackets** (десен клик върху пространстото под името на проекта и избор на **New Folder** от падащото меню):

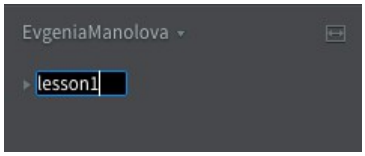
1



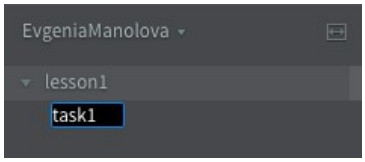
2



3



4



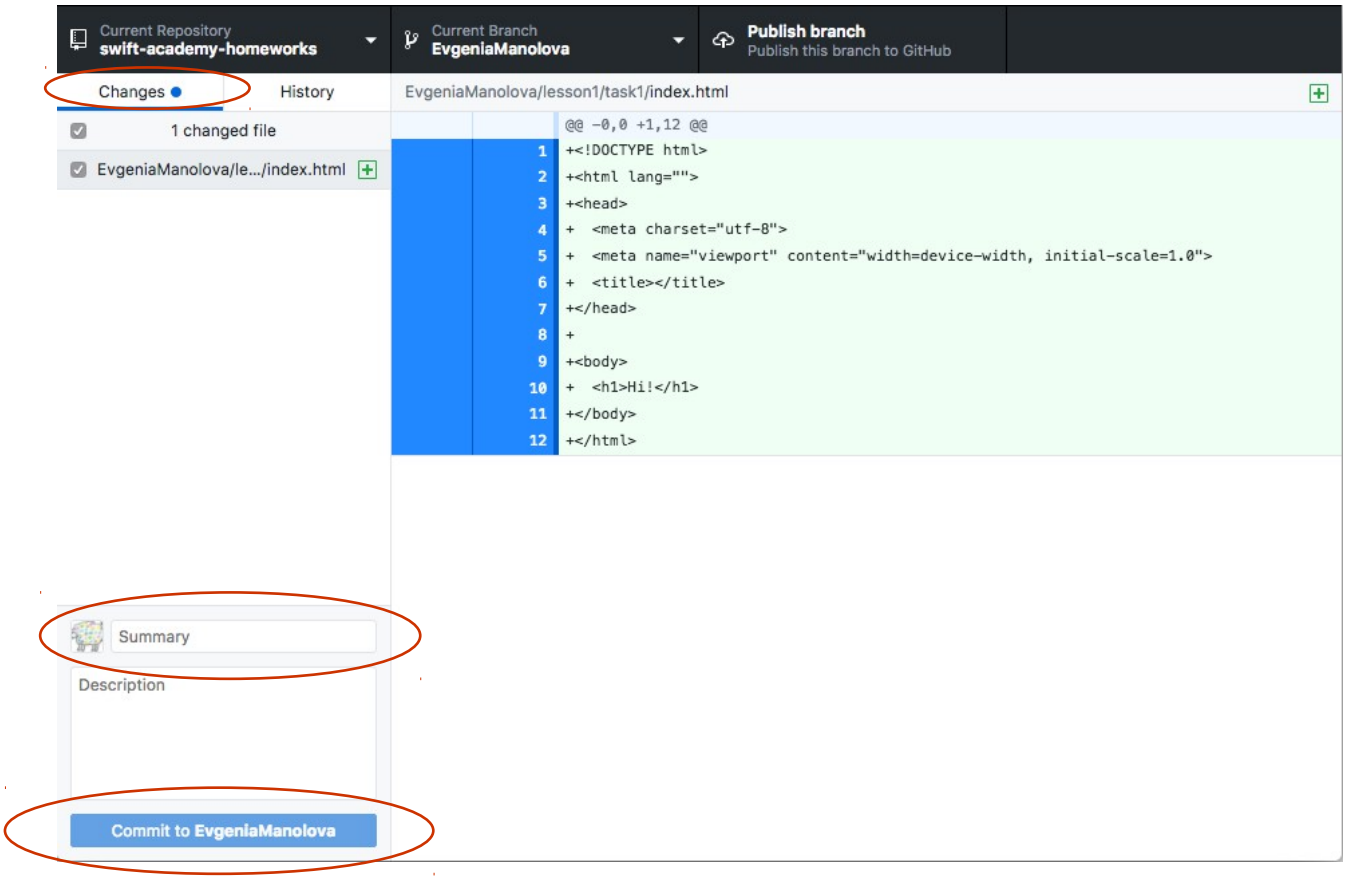
Commit

Всяка промяна, която направите в **Brackets**, веднага ще се отразява в секцията **Changes** на **Github Desktop** програмата. Тези промени, са текущи (наричат се още *локални промени*) и ако преценим, че искаме да ги запазим, трябва да направим **commit**.

Начинът, по който **Github Desktop** ни показва текущите промени, се нарича **Diff**.

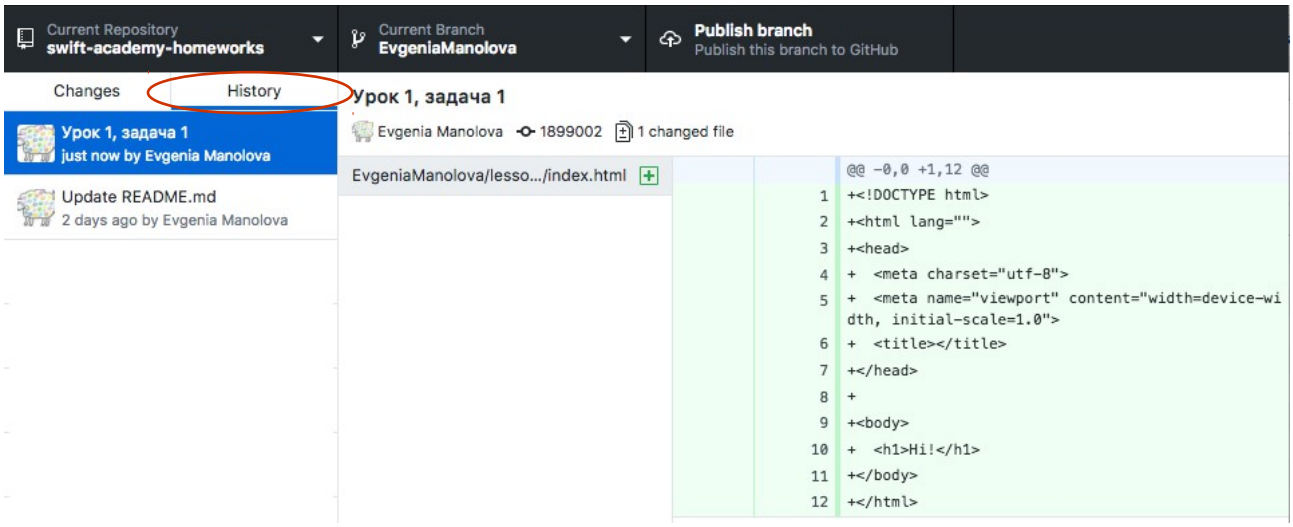
Всеки добавен нов ред се вижда със знака **‘плюс’** (+) отпред и е оцветен в зелено. Всеки премахнат ред, се показва със знака **‘минус’**(-) отпред и е оцветен в червено.

Ако преценим, че промените ни са готови за качване и предаване или просто ако искаме да си запазим текущото състояние на кода, трябва да направим **Commit**.



Commit се прави, като се попълни полето **Summary** с адекватен *commit message*. Например: "*Урок 1, задача 1*". След което се натиска бутона **Commit to BranchName**, където *BranchName* е името на вашият **branch** и в никакъв случай *не е* **master**.

В момента, в който **commit**-нем текущите си промени, те изчезват от tab-а **Changes**, и вече могат да бъдат открити в tab-а **History**:



Когато направим няколко поредни **commit**-а, чрез tab-а **History** можем да прегледаме какво точно сме били променили при всеки един от тях.

Забележка:

1. Ако само добавяме директории (папки), без да слагаме файлове в тях, те не се появяват като *текущи промени* и не могат да бъдат *commit*-нати.
2. *Commit* на промените се прави тогава, когато е постигнат някакъв етап от решението на задачата, който си заслужава да бъде запазен, защото ще се използва като основа за по-нататъшното решение.
3. *Revert* на промените се прави тогава, когато текущото решение на задачата е неправилно и *единствения* начин да продължим е като се върнем назад към състоянието на проекта от последния *commit*.

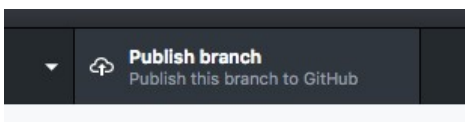
Publishing changes (Push)

Всеки един *commit* пази текущо състояние в историята на проекта.

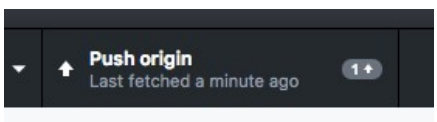
Преди да направим *Push* обаче (upload-ване на промените на сървъра), тази история съществува единствено в работното копие (*локалното repository*).

След като направим *Push*, всички *commit*-нати промени, заедно с *commit history*-то, се upload-ват на сървъра (вашето *remote repository* във вашия GitHub акаунт).

За да направим **Push**, просто трябва да натиснем бутона **Push Origin** (или **Publish Branch**, ако сме създали нов *branch*).



или

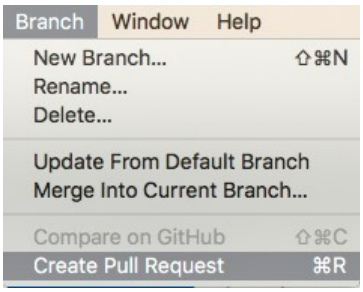


След успешен *Push*, бихме могли да видим промените си в копието на проекта в GitHub уебсайта (можем да го отворим от GitHub Desktop, чрез **Repository -> Show in GitHub**)

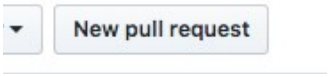
Pull request

Pull request е заявка към собственика на ***origin repository***-то (т.е. аз) да прегледа и изтегли всички промени и нови файлове от вашето *remote копие* (това, което е във вашият GitHub акаунт) във *remote origin* (това, което е в моя GitHub акаунт, т.е.: zzeni/**swift-academy-homeworks**).

Създаването на **Pull request** става или през менюто **Branch** на GitHub Desktop:



Или от уебсайта на GitHub:



Merge

Update-ването на *централното repository* (**origin**) с промените от един *Pull request*, се нарича **merge**.

Обокновенно когато се *contribute*-ва за даден проект, всички *Pull request*-и са към branch, който е различен от *master*.

Нашият проект, обаче, е доста малък и затова вашите *Pull requests* трябва да бъдат създавани към **master branch**-а.

Отделните **branch**-ове от едно repository също могат да се **merge**-ват помежду си. За това винаги трябва да проверявате дали не създавате pull request към master branch-а на вашето *собствено repository* (копието на **origin**).

Научете повече за *merge*-овете тук: https://www.youtube.com/watch?v=_ALeswWzpBo

Pull / Fetch

Когато има *remote update* (промени по файловете, направени от някой друг) в *branch*-а, в който работим, трябва да използваме функцията **Pull** на програмата GitHub Desktop, за да изтеглим тези промени в нашето работно копие.

За да разбере дали има такива *remote* промени, GitHub Desktop използва командата Fetch.