UNIVERSITÁ CATTOLICA DEL SACRO CUORE
MILANO

Interfacoltà: Economia - Scienze Matematiche, Fisiche e Naturali

MSc Innovation and Technology Management

IT coding, tools and security

Working with repositories
and
Watson Visual Recognition

Prof. Accetta Federico

Bonamici Nicole, 4809367
Dell'Anno Alessandro, 4810937

INTRODUCTION

Let imagine you are the project manager of your team and you need to coordinate, control and manage all the people who are working with you.

How will you do that?

Using GitHub is the answer.

GitHub

1. GitHub is a code hosting platform for version control and collaboration

More in detail, GitHub is a website and cloud-based service that helps developers to store and manage their code, as well as track and control changes to their code.
Before understanding exactly what GitHub is, it is necessary to explain two interconnected principles:

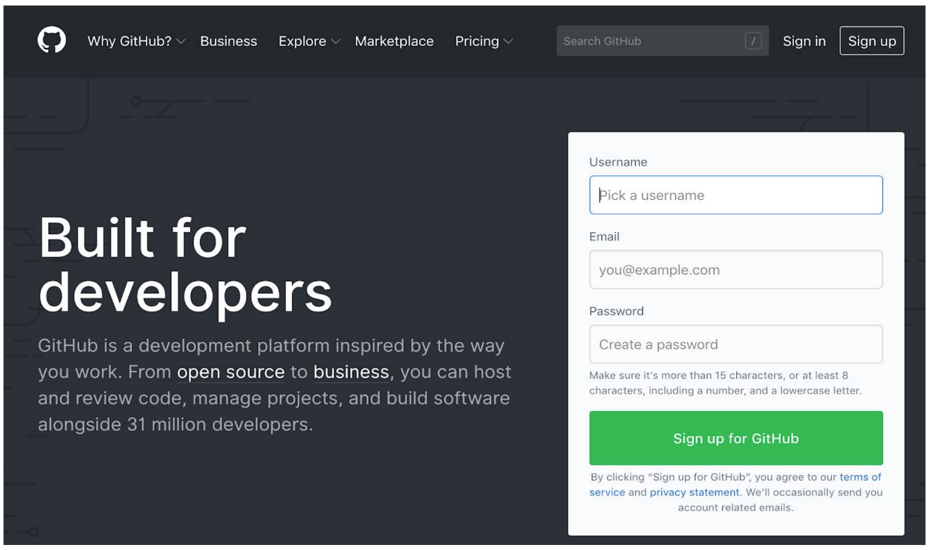
- Version control : helps developers to have multiple people working on the same projects at the same time, and many of them may be in different locations and possibly even in different countries. Everyone can know and see what everyone else is doing in real time and projects can be managed in whatever way is best for your staff and your organization's needs.
  Moreover, version control lets developers safely work through branching and merging. Firstly, the developer duplicates part of the source code (called the repository, which will be analyzed deeply later on). Then he can safely make changes to that part of the code without affecting the rest of the project. Finally, once the developer gets his part of the code working properly, he can merge that code back into the main source code to make it official.

- Git : is a specific open-source version control system created by Linus Torvalds in 2005. Specifically, Git is a distributed version control system, which means that the entire codebase and history is available on every developer's computer, which allows for easy branching and merging. According to a Stack Overflow developer survey, over 87% of developers use Git.

On June 4, 2018, Microsoft announced it had reached an agreement to acquire GitHub for US$7.5 billion. The purchase closed on October 26, 2018.

GitHub's interface is user-friendly and utterly intuitive, enough that also fledgling developers can take advantage of Git. Without GitHub, using Git generally requires a bit more technical savvy and use of the command line.
GitHub in addiction is so easy to use, that some people can even use it for other types of projects, such co-writing books.

To start, pick up a username and write down your email and password; then wait for the confirmation email to enter.
https://github.com/



Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.

Pick a username

you@example.com

Create a password

Make sure it's more than 15 characters, or at least 8 characters, including a number, and a lowercase letter.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy statement. We'll occasionally send you account related emails.

1.1 ABOUT REPOSITORIES

A repository is like a folder used to organize a single project. Your project's repository contains all of your project's files, images, videos, spreadsheets, data sets and stores each file's revision history. You can own a repository individually, and give other people collaborator access to your repository so that they can collaborate on your project. You can also share ownership of a repository with other people in an organization, and give organization members access permissions to collaborate on your repository.

Repositories can be public or private. Public repositories are visible to everyone, whereas only the owner and collaborators can view or contribute to a private repository. In order to make a repository private, you'll need to upgrade your account: this will cost $7 / month and you'll get unlimited private repositories.

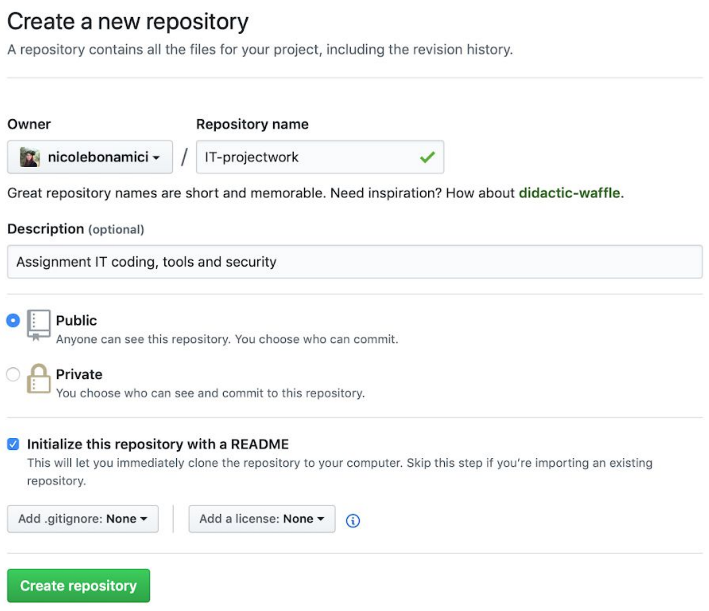
As of June 2018, GitHub reports having over 28 million users and 57 million repositories. (including 28 million public repositories), making it the largest host of source code in the world.

Each person and organization can own unlimited public repositories and invite an unlimited number of collaborators to the repository. What is important and makes github also particular is the fact every "commit".

You can collaborate on your project with others using your repository's issues, pull requests, project boards and wiki, which will be discussed further.

1.2 CREATE A NEW REPOSITORY

1. In the upper right corner, next to your avatar or identicon, click and then select New repository.
2. Name your repository.
3. Write a short description.
4. Select Initialize this repository with a README.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: nicolebonamici / Repository name: IT-projectwork

Great repository names are short and memorable. Need inspiration? How about didactic-waffle.

Description (optional): Assignment IT coding, tools and security

● Public — Anyone can see this repository. You choose who can commit.

○ Private — You choose who can see and commit to this repository.

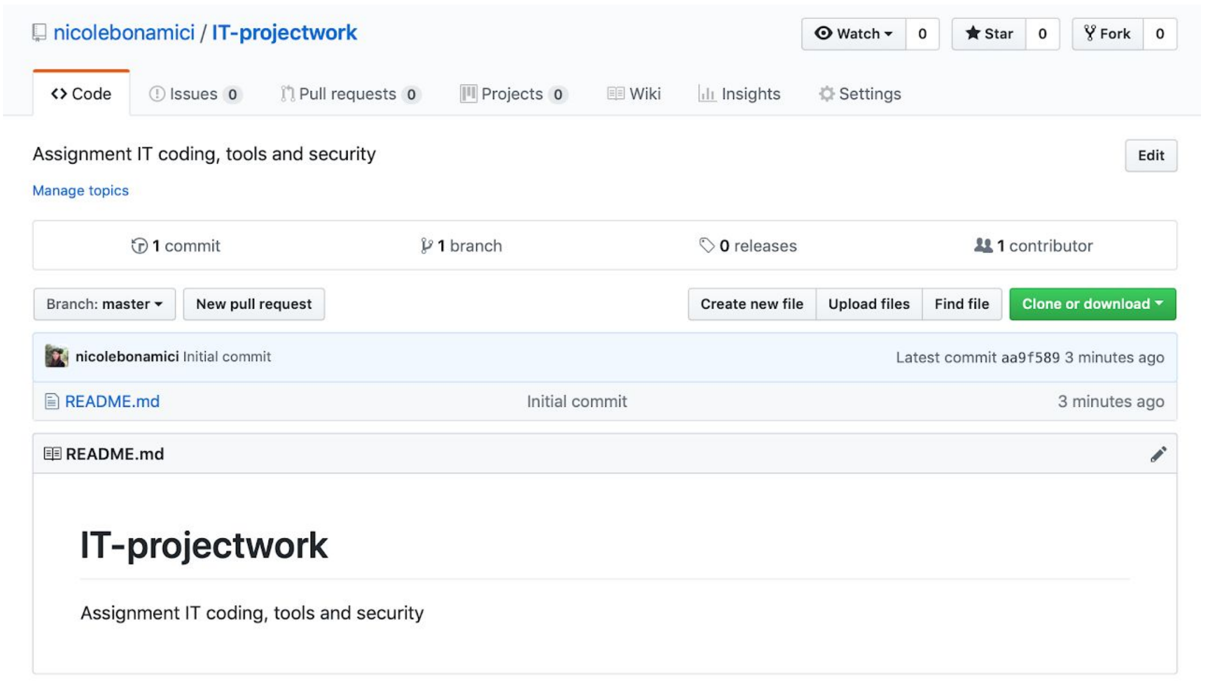
☑ Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

https://github.com/nicolebonamici/IT-projectwork

1.3 EXPLORING THE GITHUB INTERFACE



nicolebonamici / IT-projectwork

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Assignment IT coding, tools and security

Manage topics

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

nicolebonamici Initial commit — Latest commit aa9f589 3 minutes ago

README.md — Initial commit — 3 minutes ago

README.md

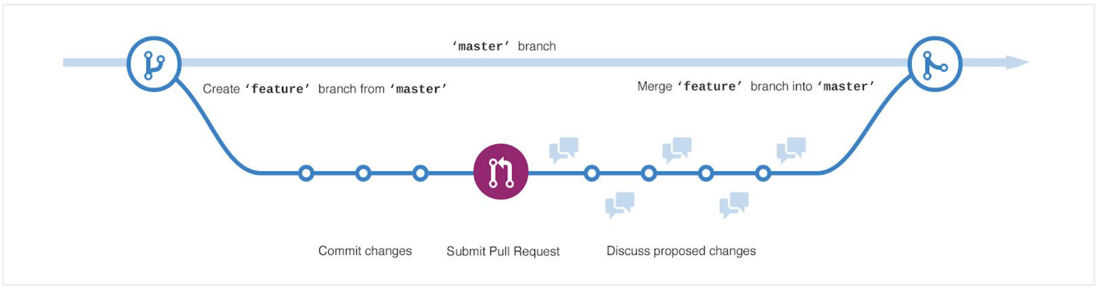
IT-projectwork

Assignment IT coding, tools and security

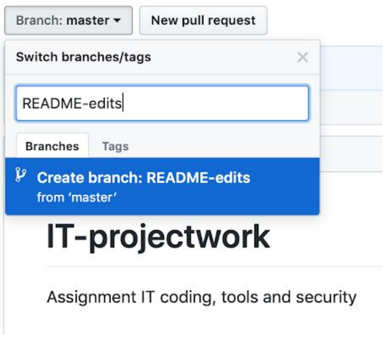
1.4 CREATE A BRANCH

Branching is the way to work on different versions of a repository at one time.
By default your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.
When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.
This diagram shows:

- The master branch
- A new branch called feature (because we're doing 'feature work' on this branch)
- The journey that feature takes before it's merged into master


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

Developers, writers, and designers use branches for keeping bug fixes and feature work separate from our master branch. When a change is ready, they merge their branch into master.



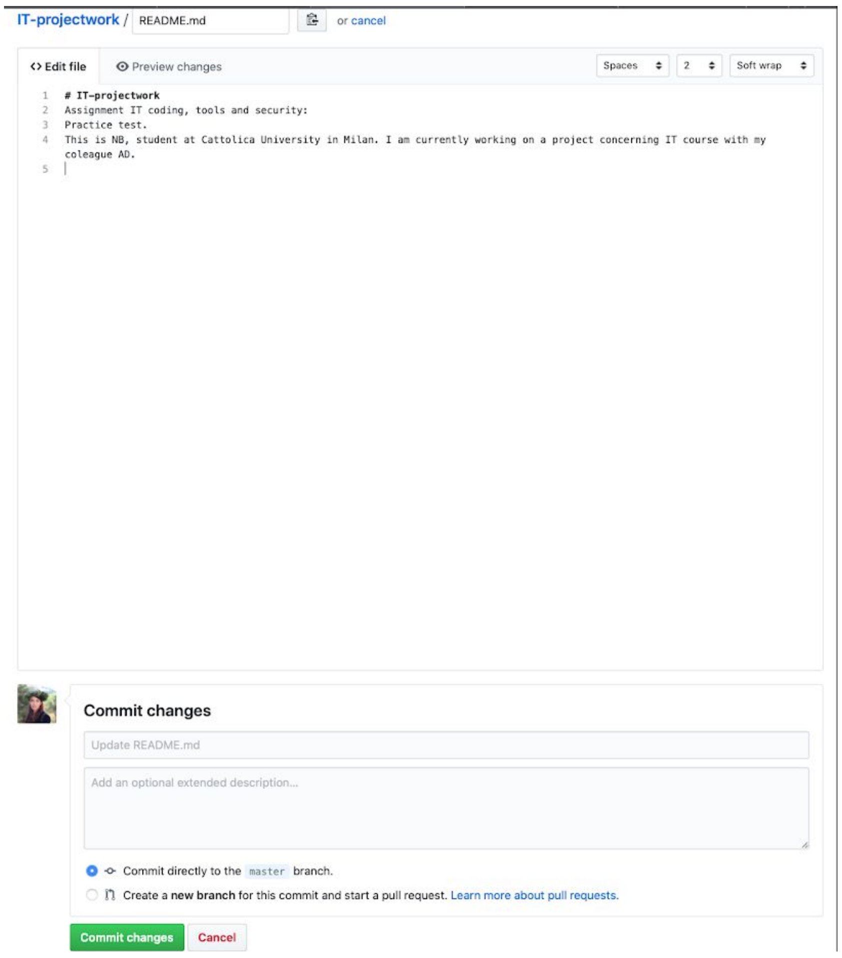
Go to your new repository IT-projectwork.

1. Click the drop down at the top of the file list that says **branch: master**.
2. Type a branch name, README-edits, into the new branch text box.
3. Select the blue **Create branch** box or hit "Enter" on your keyboard.

Now you have two branches, master and readme-edits. They look exactly the same, but not for long! Next we'll add our changes to the new branch.

1.5 MAKE AND COMMIT CHANGES

On GitHub, saved changes are called **commits**. Each commit has an associated commit *message*, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you've done and why.


IT-projectwork / README.md
or cancel
Edit file
Preview changes
Spaces
2
Soft wrap

```
# IT-projectwork
Assignment IT coding, tools and security:
Practice test.
This is NB, student at Cattolica University in Milan. I am currently working on a project concerning IT course with my
coleague AD.
```

Commit changes
Update README.md
Add an optional extended description...
Commit directly to the master branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit changes
Cancel

How to make and commit changes:

1. Click the README.md file.
2. Click the pencil icon in the upper right corner of the file view to edit.
3. In the editor, write a bit about yourself.
4. Write a commit message that describes your changes.
5. Click Commit changes button.

These changes will be made to just the README file on your readme-edits branch, so now this branch contains content that's different from master.

1.6 OPEN A PULL REQUEST

Now that you have changes in a branch off of master, you can open a **pull request**.

When you open a **pull request**, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.
By using GitHub’s @mention system in your pull request message, you can ask for feedback from specific people or teams.
You can even open pull requests in your own repository and merge them yourself.

1.7 MERGE YOUR PULL REQUEST

In this final step, it’s time to bring your changes together – merging your `readme-edits` branch into the `master` branch.

1. Click the green **Merge pull request** button to merge the changes into `master`.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.

1.8 WORKING WITH ISSUES

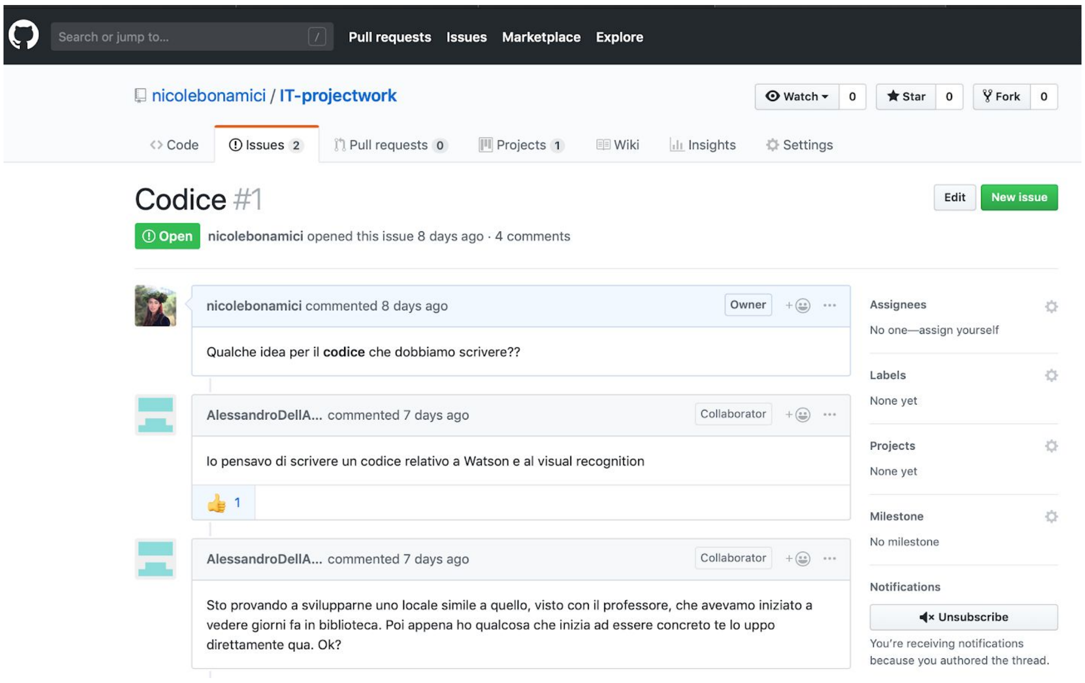
You can use issues to track ideas, enhancements, tasks, or bugs for work on GitHub.

You can collect user feedback, report software bugs, and organize tasks you'd like to accomplish with issues in a repository. Issues can act as more than just a place to report software bugs. Imagine that you are working in a big group of developers, where everyone has to fill his tasks (which are different functionality of the code). It always happens that many bugs occur during the project work. In this case we may exploit github potential and highline where the mistakes are.The step we can take into consideration is that we can renamed an issue (I.E “ completa 1”) and assign that specific issue to a specific person of the group. Once we finished our tasks we can end it just using the “closing issue” for a repository and renamed that issue so that the manager of the group will know that the mission is concluded.

To stay updated on the most recent comments in an issue, you can watch an issue to receive notifications about the latest comments.

With issues, you can:

- Track and prioritize your work using project boards.
- Associate issues with pull requests so that your issue automatically closes when you merge a pull request.
- Create issue templates to help contributors open meaningful issues.
- Transfer open issues to other repositories.
- Track duplicate issues using saved replies.
- Report comments that violate GitHub's Community Guidelines.



Codice #1

Open nicolebonamici opened this issue 8 days ago · 4 comments

nicolebonamici commented 8 days ago (Owner)

Qualche idea per il **codice** che dobbiamo scrivere??

AlessandroDellA... commented 7 days ago (Collaborator)

Io pensavo di scrivere un codice relativo a Watson e al visual recognition

👍 1

AlessandroDellA... commented 7 days ago (Collaborator)

Sto provando a svilupparne uno locale simile a quello, visto con il professore, che avevamo iniziato a vedere giorni fa in biblioteca. Poi appena ho qualcosa che inizia ad essere concreto te lo uppo direttamente qua. Ok?

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Notifications: Unsubscribe

You're receiving notifications because you authored the thread.

1.9 PULL REQUESTS

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.You can add a summary of the proposed changes, review the changes made by commits, add labels, milestones, and assignees, and @mention individual contributors or teams. So, as you might have understood this is the tool which we can use to put together our branches once they have splitted up from each other.

1.10 PROJECT BOARDS

Project boards on GitHub help you organize and prioritize your work. You can create project boards for specific feature work, comprehensive roadmaps, or even release checklists. With project boards, you have the flexibility to create customized workflows that suit your needs.

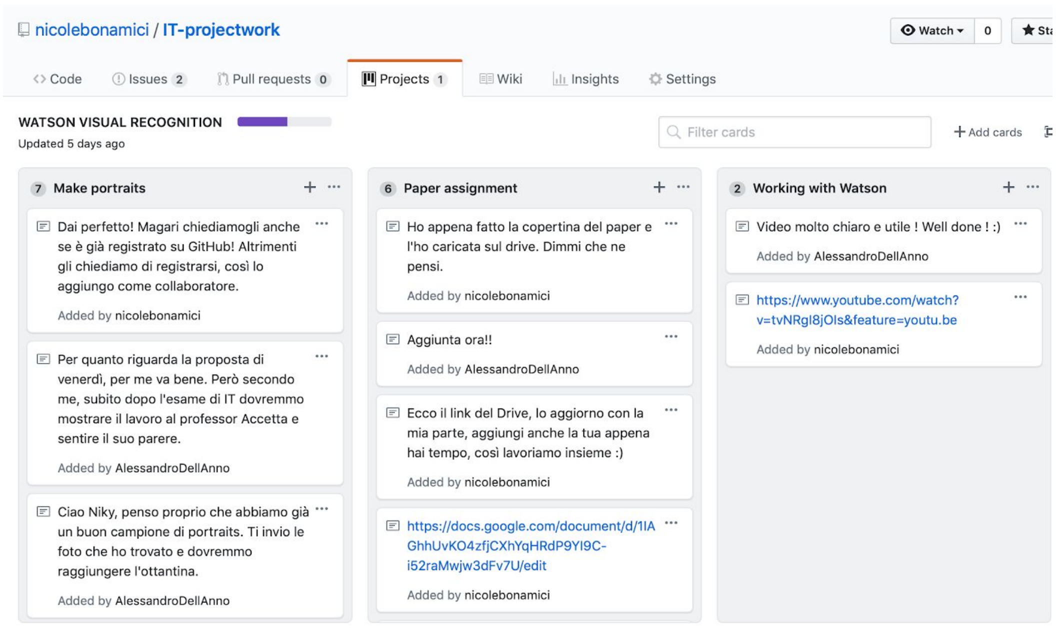
Project boards are made up of issues, pull requests, and notes that are categorized as cards in columns of your choosing. You can drag and drop or use keyboard shortcuts to reorder cards within a column, move cards from column to column, and change the order of columns.

Project board cards contain relevant metadata for issues and pull requests, like labels, assignees, the status, and who opened it. You can view and make lightweight edits to issues and pull requests within your project board by clicking on the issue or pull request title.

You can create notes within columns to serve as task reminders, references to issues and pull requests from any repository on GitHub, or to add information related to the project board. You can create a reference card for another project board by adding a link to a note. If the note isn't sufficient for your needs, you can convert it to an issue. For more information on converting project board notes to issues, see "Adding notes to a project board."

There are two types of project boards:

- **Repository project boards** are scoped to issues and pull requests within a single repository. They can also include notes that reference issues and pull requests in other repositories.
- **Organization-wide project boards** can contain issues and pull requests from any repository that belongs to an organization. You can link up to five repositories to your organization project board to make it easier to add issues and pull requests from those repositories to your project board. For more information, see "Linking a repository to a project board."



WATSON VISUAL RECOGNITION
Updated 5 days ago

7 Make portraits

Dai perfetto! Magari chiediamogli anche se è già registrato su GitHub! Altrimenti gli chiediamo di registrarsi, così lo aggiungo come collaboratore.
Added by nicolebonamici

Per quanto riguarda la proposta di venerdì, per me va bene. Però secondo me, subito dopo l'esame di IT dovremmo mostrare il lavoro al professor Accetta e sentire il suo parere.
Added by AlessandroDellAnno

Ciao Niky, penso proprio che abbiamo già un buon campione di portraits. Ti invio le foto che ho trovato e dovremmo raggiungere l'ottantina.
Added by AlessandroDellAnno

6 Paper assignment

Ho appena fatto la copertina del paper e l'ho caricata sul drive. Dimmi che ne pensi.
Added by nicolebonamici

Aggiunta ora!!
Added by AlessandroDellAnno

Ecco il link del Drive, lo aggiorno con la mia parte, aggiungi anche la tua appena hai tempo, così lavoriamo insieme :)
Added by nicolebonamici

https://docs.google.com/document/d/1IAGhhUvKO4zfjCXhYqHRdP9YI9C-i52raMwjw3dFv7U/edit
Added by nicolebonamici

2 Working with Watson

Video molto chiaro e utile ! Well done ! :)
Added by AlessandroDellAnno

https://www.youtube.com/watch?v=tvNRgI8jOIs&feature=youtu.be
Added by nicolebonamici

1.11 WIKI

Every GitHub repository comes equipped with a section for hosting documentation, called a wiki.
Basically, it works like Wikipedia, meaning that you can share long-form content about your project, such as how to use it, how it's been designed, manifestos on its core principles, and so on.
Wikis, exactly as commits, can be edited directly on GitHub or you can work with a text editor offline (like *Visual Studio Code* as will be explained further) and simply push your changes. Wikis are collaborative by design. By default, only collaborators on your repository can make changes to wikis, but you can configure this to be enabled for all users on public repositories.

https://github.com/nicolebonamici/IT-projectwork/wiki/Guide-Instruction

2. Visual Studio Code has a high productivity code editor which, when combined with programming language services, gives you the power of an IDE and the speed of a text editor

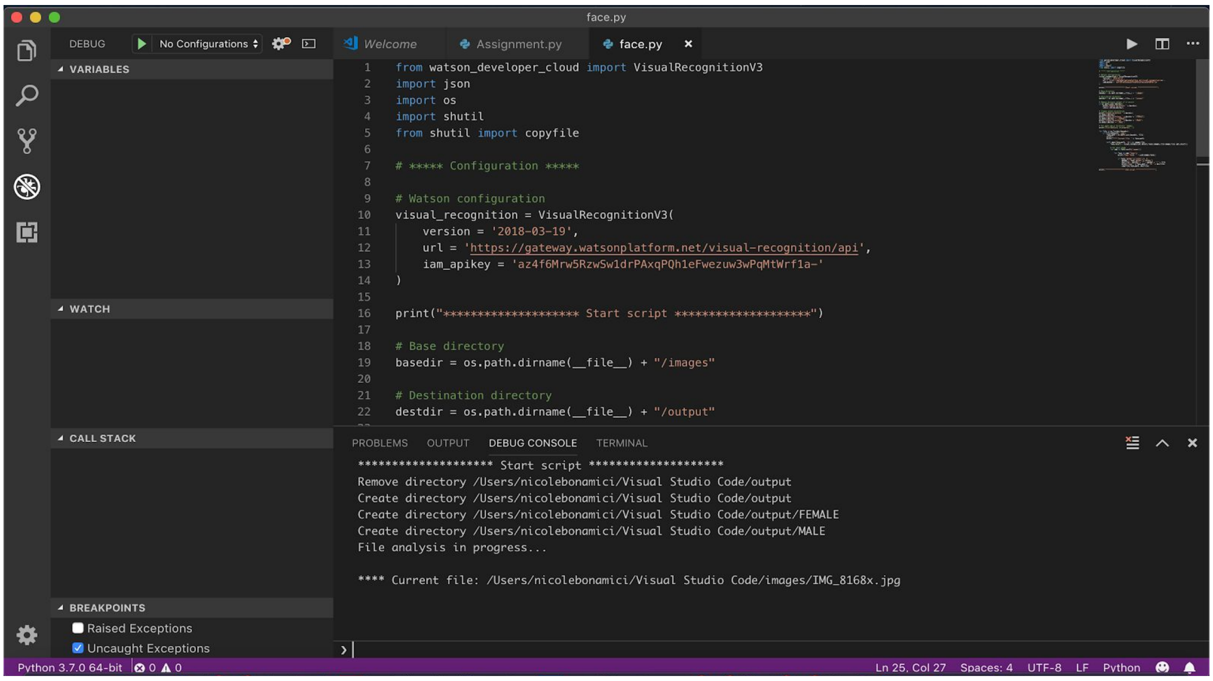
Visual Studio Code is a source code editor created by Microsoft and it is available for Windows, MacOs and Linux. Visual Studio Code was announced on April 29, 2015 by Microsoft at the 2015 Build conference.
In the Stack Overflow 2018 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 34.9% of 75,398 respondents claiming to use it.

It includes support for debugging (with the term "debugging" we represent the activity that identifies and then correct the so called "bugs", mistakes in writing some codes), embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.
It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences. It is free and open-source, although the official download is under a proprietary license.

To sum up it is something in between IDE (integrated development environment) and an editor. It can be used for Node.js projects and also ASP.NET (as his predecessors "virtual studio").

VS Code comes with a simple and intuitive layout that maximizes the space provided for the editor while leaving ample room to browse and access the full context of your folder or project. It is divided into five areas:

- Editor - The main area to edit your files. You can open as many editors as you like side by side vertically and horizontally.
- Side Bar - Contains different views like the Explorer to assist you while working on your project.
- Status Bar - Information about the opened project and the files you edit.
- Activity Bar - Located on the far left-hand side, this lets you switch between views and gives you additional context-specific indicators, like the number of outgoing changes when Git is enabled.
- Panels - You can display different panels below the editor region for output or debug information, errors and warnings, or an integrated terminal. Panel can also be moved to the right for more vertical space



```python
from watson_developer_cloud import VisualRecognitionV3
import json
import os
import shutil
from shutil import copyfile

# ***** Configuration *****

# Watson configuration
visual_recognition = VisualRecognitionV3(
    version = '2018-03-19',
    url = 'https://gateway.watsonplatform.net/visual-recognition/api',
    iam_apikey = 'az4f6Mrw5RzwSw1drPAxqPQh1eFwezuw3wPqMtWrf1a-'
)

print("******************** Start script ********************")

# Base directory
basedir = os.path.dirname(__file__) + "/images"

# Destination directory
destdir = os.path.dirname(__file__) + "/output"
```

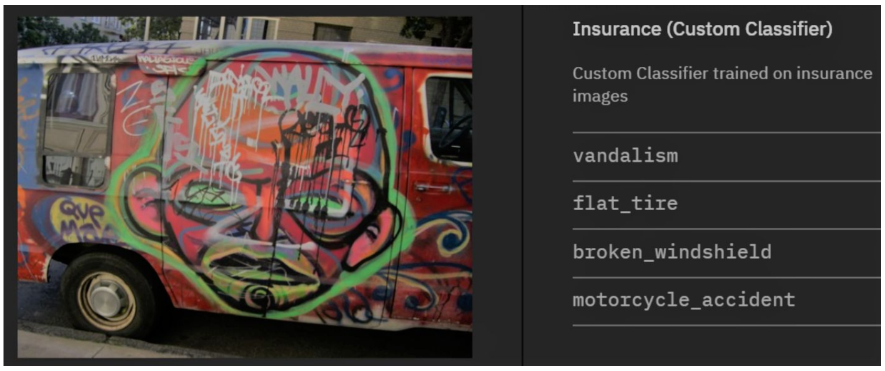
```
******************** Start script ********************
Remove directory /Users/nicolebonamici/Visual Studio Code/output
Create directory /Users/nicolebonamici/Visual Studio Code/output
Create directory /Users/nicolebonamici/Visual Studio Code/output/FEMALE
Create directory /Users/nicolebonamici/Visual Studio Code/output/MALE
File analysis in progress...

**** Current file: /Users/nicolebonamici/Visual Studio Code/images/IMG_8168x.jpg
```

3. Watson Visual Recognition understands an image's content out-of-the-box

Visual Recognition understands the contents of images. Analyze images for scenes, objects, faces, colors, food, text, explicit content and other subjects that can give you insights into your visual content. You can customize Watson perfectly for your unique use case. With only a few images, Watson can learn any new object, person, or attribute. One example to understand this can be found in the automotive industry. We are talking about the co-work with IBM and Autoglass® BodyRepair. The latter implements IBM Watson visual recognition to its offer in order to eliminate and avoid the human need. In fact, Watson can assess customers' vehicle damage and automatically recommends a repair price.

Step 1: identifying different damages (if there is any kind of damage)



Insurance (Custom Classifier)

Custom Classifier trained on insurance images

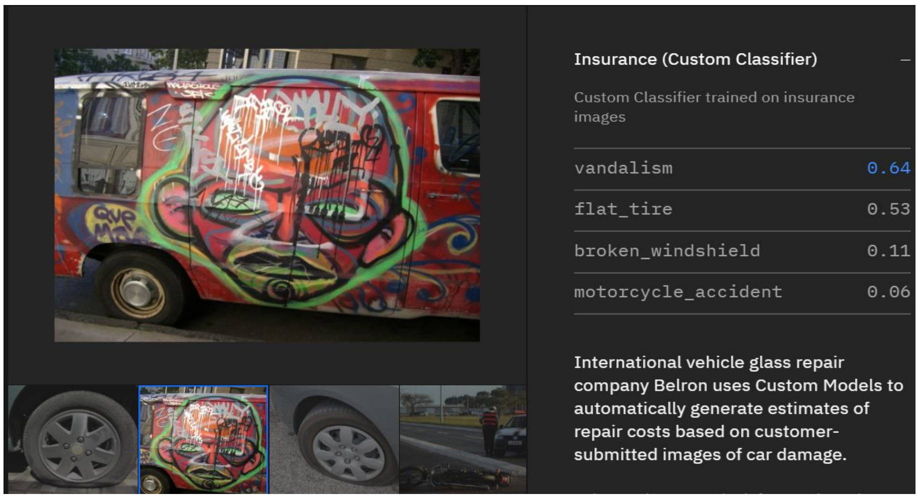
vandalism

flat_tire

broken_windshield

motorcycle_accident

Step 2: Evaluate damage



Insurance (Custom Classifier)

Custom Classifier trained on insurance images

vandalism	0.64
flat_tire	0.53
broken_windshield	0.11
motorcycle_accident	0.06

International vehicle glass repair company Belron uses Custom Models to automatically generate estimates of repair costs based on customer-submitted images of car damage.

Which are the results?

- >70% improvement in quote processing times

  by using AI analyses to determine repair costs
- Optimizes damage advisors' time and skills

  by allowing employees to focus on more complex claims
- Boosts efficiency of the quote generation process

  by supporting human decision-making with AI analyses

3.1. PRACTICAL CASE (while reading, please have a look at the same time at the code. We tried to explain what we did)

Working with repositories means also take someone's else work and modify it depending on your needs. This is what we did by writing and using "from Watson_developer_cloud"and then we decided to import "visual recognition" because it was the tool we needed it.

//Here the link where we have started to get the idea:

https://github.com/watson-developer-cloud/python-sdk/blob/master/examples/visual_recognition_v3.py//

Import *Json* (it stands for Javascript Object notation, is a very widely used format type for data exchange in server client applications such as APIs or Mashup. It is based on JavaScript, but its development is specific to the exchange of data and is independent of the development of the scripting language from which it is born and with which it is perfectly integrated and simple to use). So, since we are using Watson, we are going to interact with it through Json.

Import *Os* is a library that provides the possibility to interact with the operating system. It contains all the functions, such as creating a folder, copy, paste it and so on. In this case we used it to paste every input into the folder called "output" (divided into female and male folders)

Import *Shutil* is a library that provides the possibility to manage different file lists (for each file) in that folder, do a particular thing and so on.

If face["gender"]["score"] > 0.5: we did this in order to skimming and take only appropriable results.

An application programming interface key (API key) is a unique code that is passed in to an API to identify the calling application or user. API keys are used to track and control how the API is being used, for example to prevent malicious use or abuse of the API. The API key often acts as both a unique identifier and a secret token for authentication, and generally has a set of access that is specific to the identity associated with it.

Since we have created a public repository (because the private one was not free), we have decided not to upload our private API key.

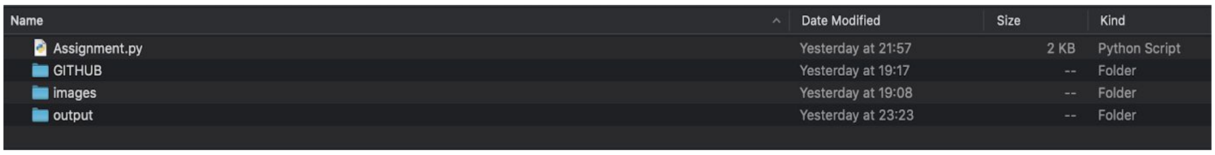
It is important that the user writes its own API key in order to run the application.

Basedir is the base directory and shows you where Watson is going to take what it needs. It uses the library Operating System (os) and it asks to get the name of the file where there is the folder "image".

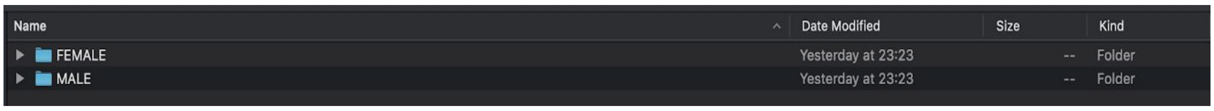
Destdir is the destination directory where all the pictures from "image" will be located and splitted in gender. It is used to define which is the folder where all the outputs will be created.

If the directory output already exists, we have asked Python to delete it, just to mob it up.

Then inside the directory output we want to create two different folders one for male and one for female in order to have all the portraits splitted up in gender.



Name	Date Modified	Size	Kind
Assignment.py	Yesterday at 21:57	2 KB	Python Script
GITHUB	Yesterday at 19:17	--	Folder
images	Yesterday at 19:08	--	Folder
output	Yesterday at 23:23	--	Folder



Name	Date Modified	Size	Kind
FEMALE	Yesterday at 23:23	--	Folder
MALE	Yesterday at 23:23	--	Folder

We have started a big for-each loop that iterates through each element: if the file analyzed is a *.JPG* file, the application will keep the absolute name of the picture. We ask each file to be opened and analyzed by Watson

"*with open(face_path, 'rb') as image_file*": It starts to take into account the first image and just after that, activating the watson library, we will have the output in json file. So now we have our final object but as the last thing to do, we need to use it.

Through the "*for-each*", we iterates all the images in the inputs folder and we let the system says how many faces there were in each photo. In particular, with "*print(json)*" we receive our desired output which is the info watson is able to detect from the face. We, in fact, can detect many different info such as "*age range*" and "*male or female*". REMEMBER! We are going to receive this info in *json* type. As we said before we put " > *0.5*" in order to skimming and take only appropriable results.

Once the photo has been analysed, Watson will put the photo into the right folder. This process is for one image, and watson will go back to the first "*for-each*"(line 32) and will do the same operations with all the other images. (MALE/FEMALE)

CONCLUSION

Repositories are shaping our future.
It is a practice that it is started to take root in many enterprises.
Working with repositories allows manager and developers to share a common goal.
It is easy and intuitive and it is a new way of sharing information and ideas.

Our purpose, as students, was only to understand how repositories work in order to take stock of it, hoping that all this work will be helpful for our future job.