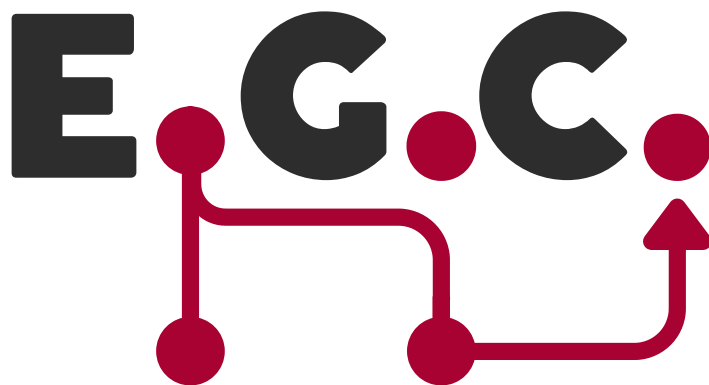




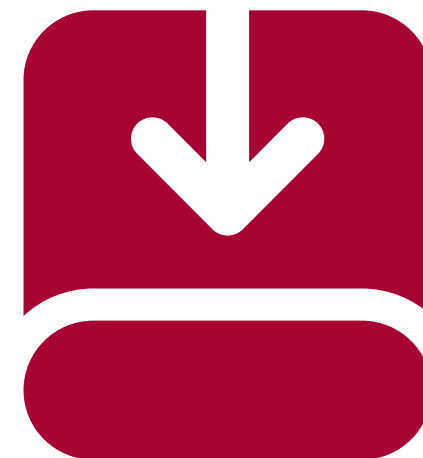
Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración



Escuela Técnica Superior de
Ingeniería Informática

Práctica 1 Instalación del sistema base

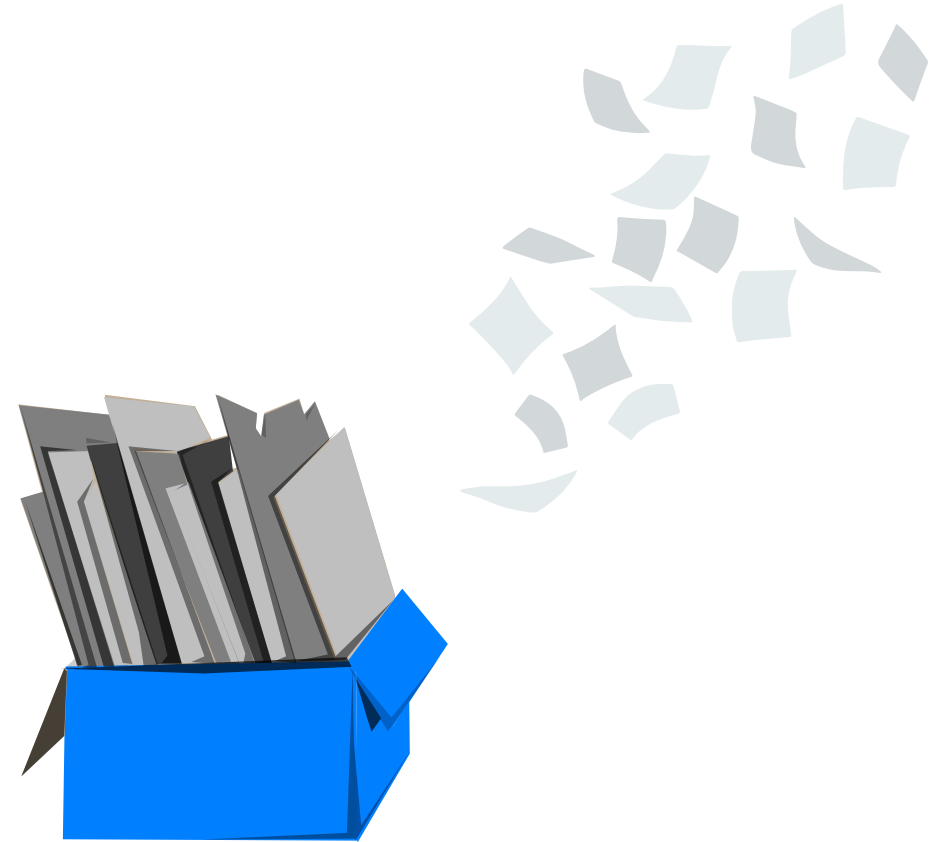


1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

1. **¿Qué es uvlhub.io?**
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

1. ¿Qué es uvlhub.io?

Dataset en ciencia e investigación



¡Muchos datasets!

1. ¿Qué es uvlhub.io?

Dataset en ciencia e investigación



- Datasets observacionales
- Datasets experimentales
- Datasets de encuestas
- Datasets genómicos
- Datasets de imágenes
- Datasets de series temporales
- Datasets geoespaciales
- Datasets de simulación
- ...

1. ¿Qué es uvlhub.io?

Dataset en ciencia e investigación



Son IMPORTANTES porque permiten...

- Base empírica
- Reproducibilidad
- Nuevos hallazgos
- Colaboración
- Transparencia
- Educación

1. ¿Qué es uvlhub.io?

El problema



**Repositorios
privados**



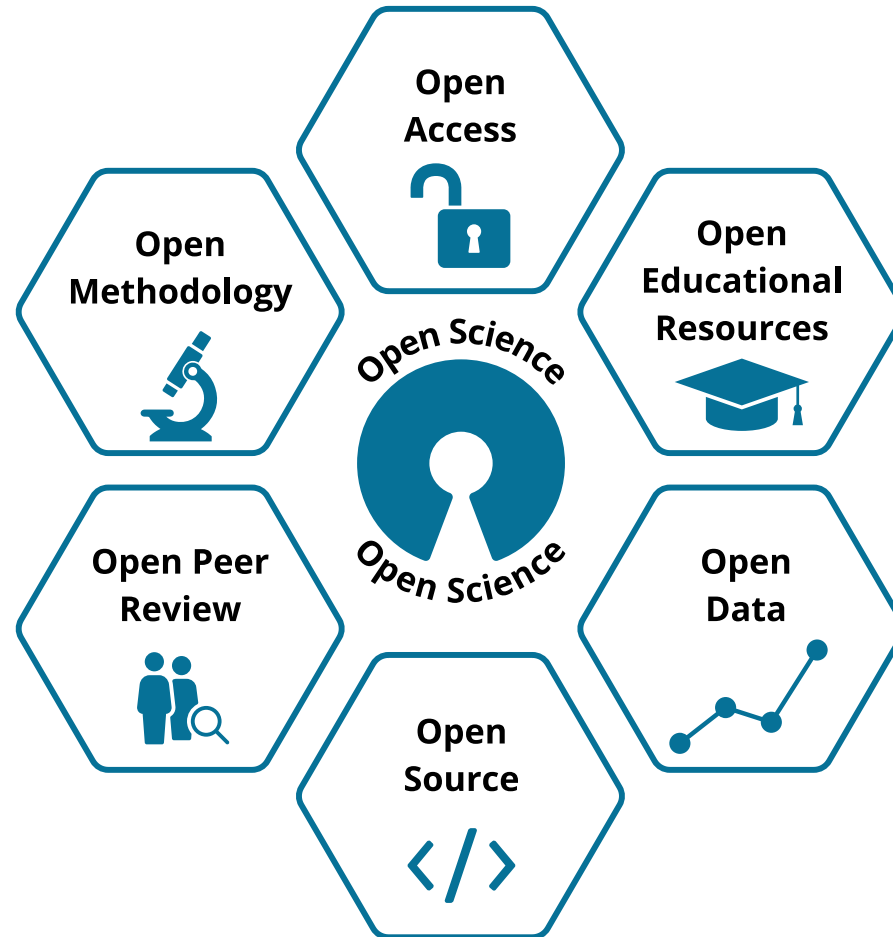
**Datasets perdidos,
corruptos,
incompletos...**



**¡Sistema
caído!**

1. ¿Qué es uvlhub.io?

Los seis principios de la Ciencia Abierta



1. ¿Qué es uvlhub.io?

La solución



**repositorio de modelos de características en formato UVL
siguiendo principios de la ciencia abierta**



UNIVERSIDAD
DE MÁLAGA



universität
uulm

1. ¿Qué es uvlhub.io?



Repositorio oficial para la asignatura

github.com/EGCETSII/uvlhub

1. ¿Qué es uvlhub.io?
 2. **Documentación oficial**
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

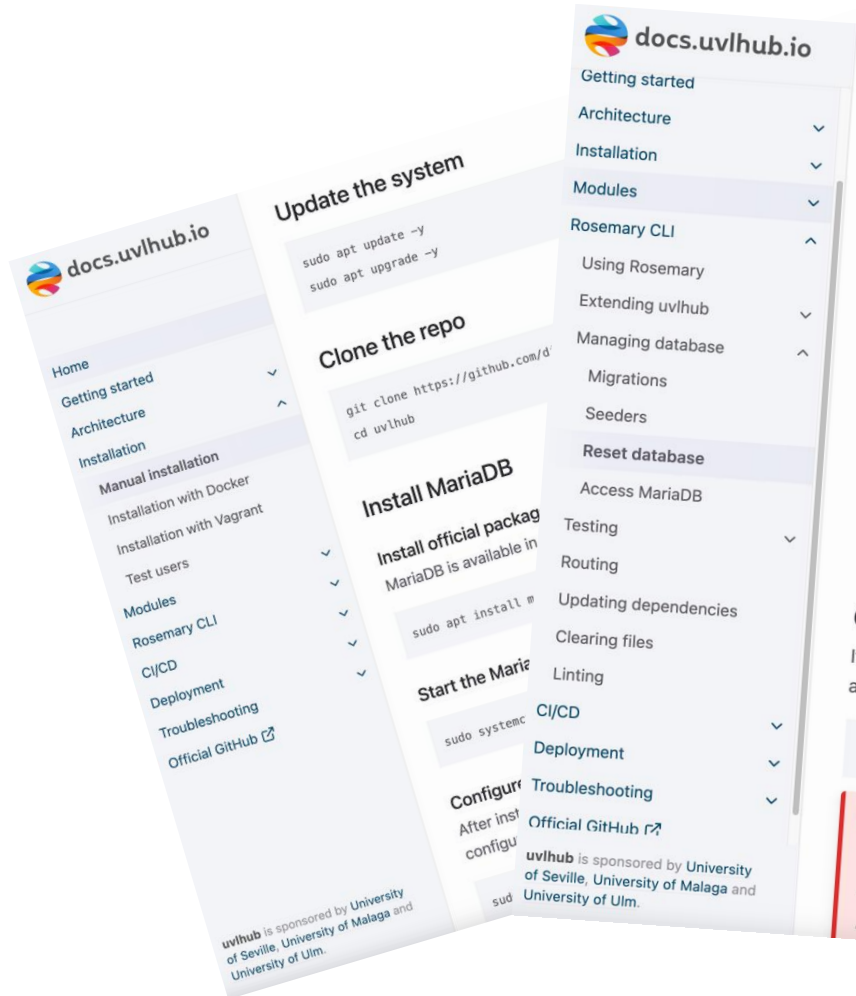
2. Documentación oficial



docs.uvhub.io

guías de instalación, guías de comandos,
migraciones, despliegue, módulos,
resolución de dudas...

2. Documentación oficial



Reset database

The `rosemary db:reset` command is a powerful tool for resetting your project's database to its initial state. This command deletes all the data in your database, making it ideal for fixing any inconsistencies we may have created during development.

TABLE OF CONTENTS

- 1 Basic Usage
- 2 Clear migrations

Basic Usage

To reset your database and clear all table data except for migration records, run:

```
rosemary db:reset
```

The `rosemary db:reset` command also clears the uploads directory as part of the reset process, ensuring that any files uploaded during development or testing are removed.

Clear migrations

If you need to completely rebuild your database from scratch, including removing all migration history and starting fresh, you can use the `--clear-migrations` option:

```
rosemary db:reset --clear-migrations
```

BE CAREFUL! THIS COMMAND WILL...

- Delete all data from the database, including the migration history.
- Clear the migrations directory.
- Initialize a new set of migrations.

Part two: deploy application in Render

We are going to use Render as a cloud deployment service.

... that we use our GitHub account because it will be easier to link the working.

specific configuration (install dependencies, scripts, create migrations, our own Dockerfile image for the Render service.

Click on `Dashboard` and then on the `New Web Service` button.

Build and deploy from a `Git repository`.

repository where we are working. Click on the `Connect` button.

It `uvlhub_web_app` or similar, any available name is fine.

choose `Frankfurt (central EU)`.

... , unless we have a reason to do so, it should be `main`.

`Directory`, we leave it blank.

we choose `Docker`.

: `Type`, we choose `Free`.

ENVIRONMENT VARIABLES CONFIGURATION.

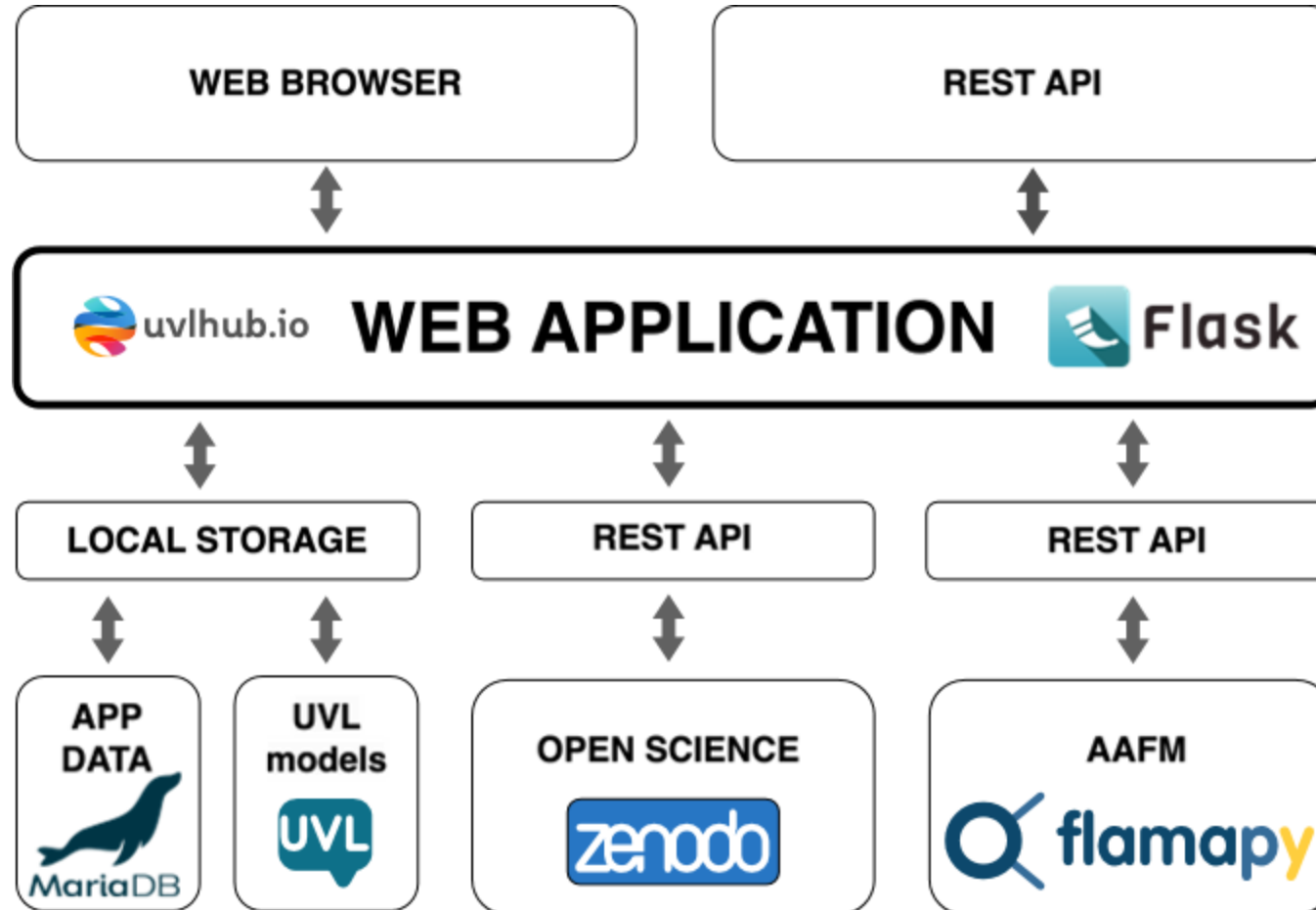
Environment variables, to avoid defining each environment variable one by one, click on `Add from` and paste this:

```
AE="UVLHUB.IO"
roduction
er.com
```

1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. **Arquitectura de la aplicación**
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

3. Arquitectura de la aplicación

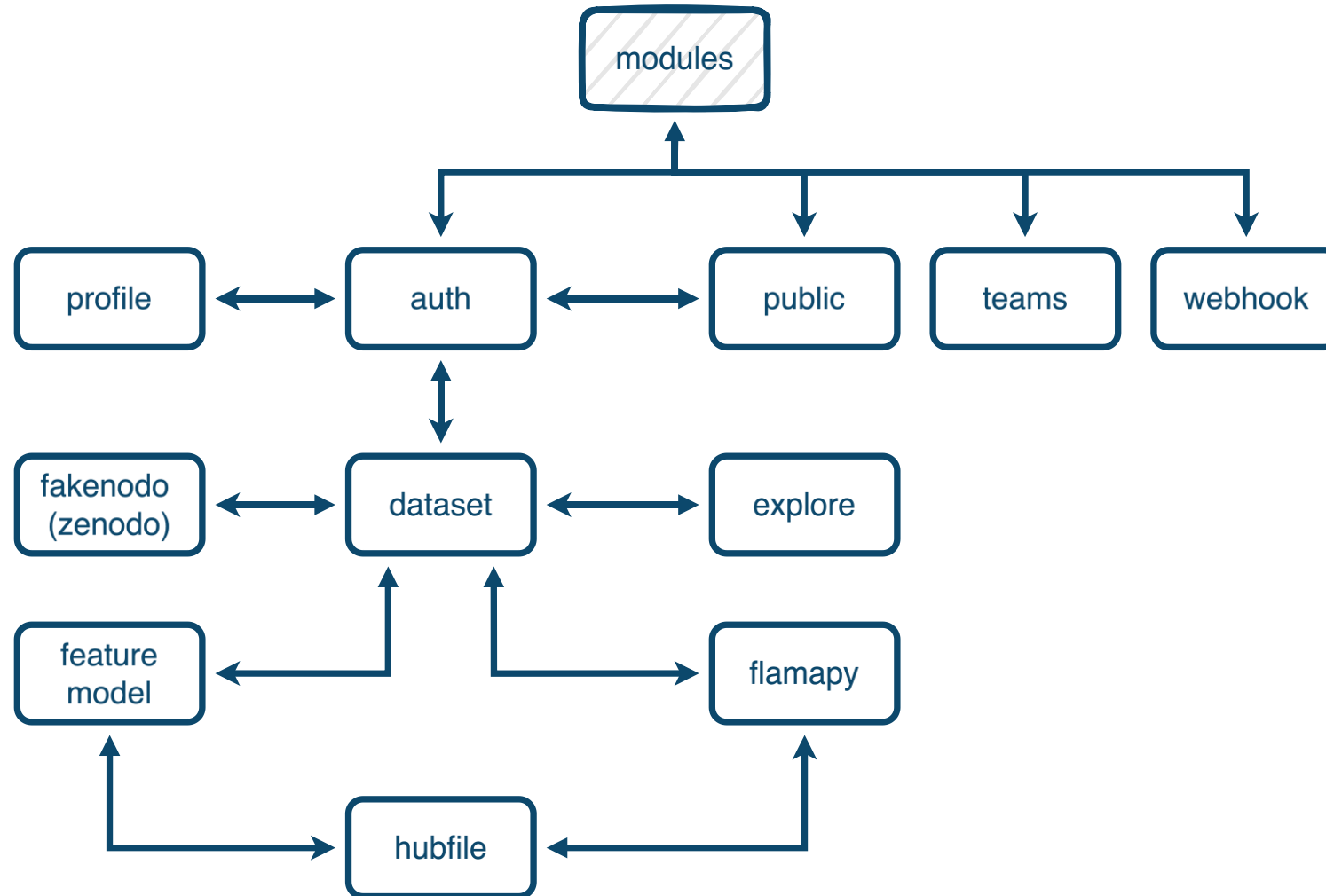
Vistazo general



docs.uvlhub.io/architecture/overview

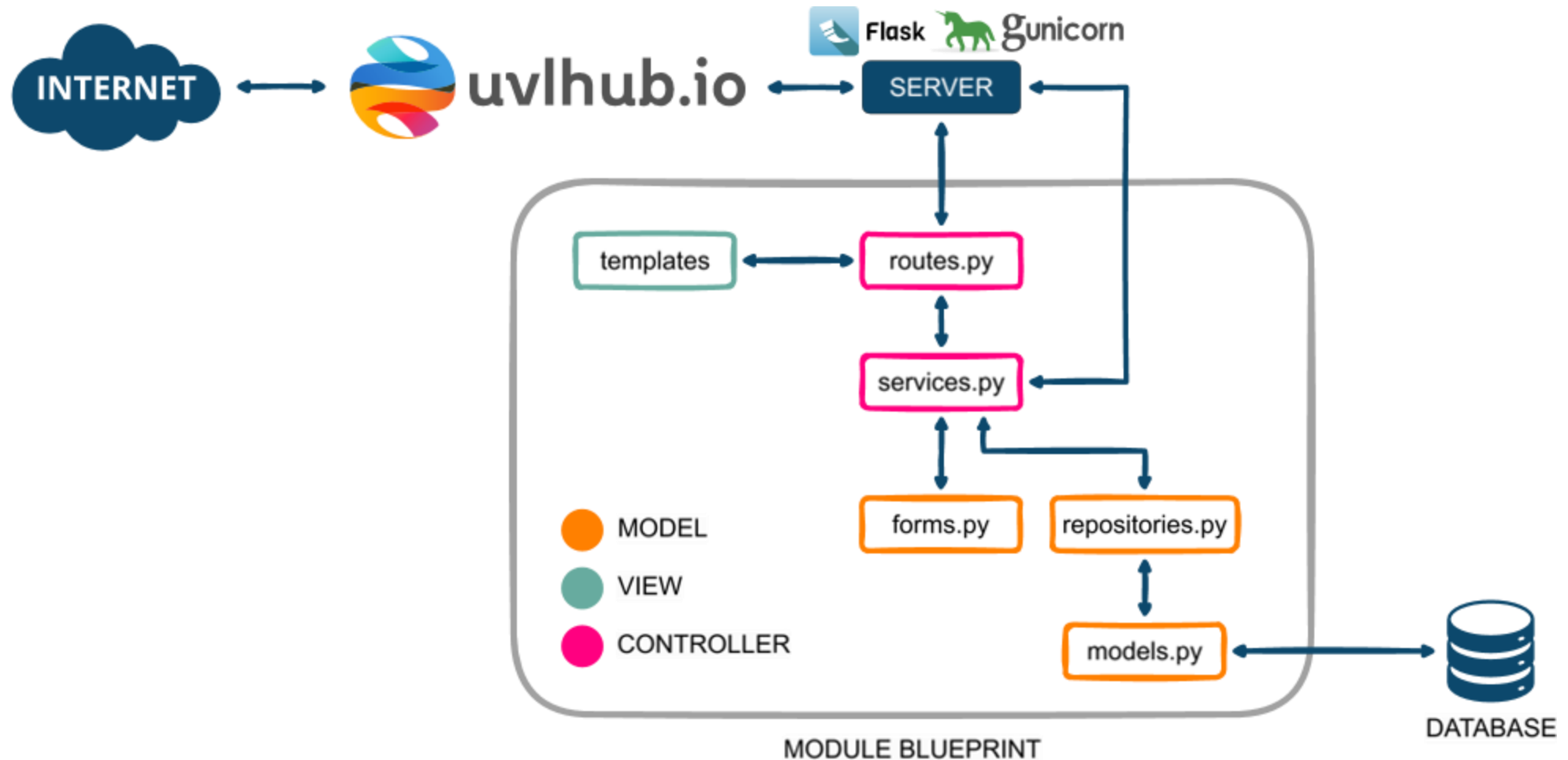
3. Arquitectura de la aplicación

Módulos



3. Arquitectura de la aplicación

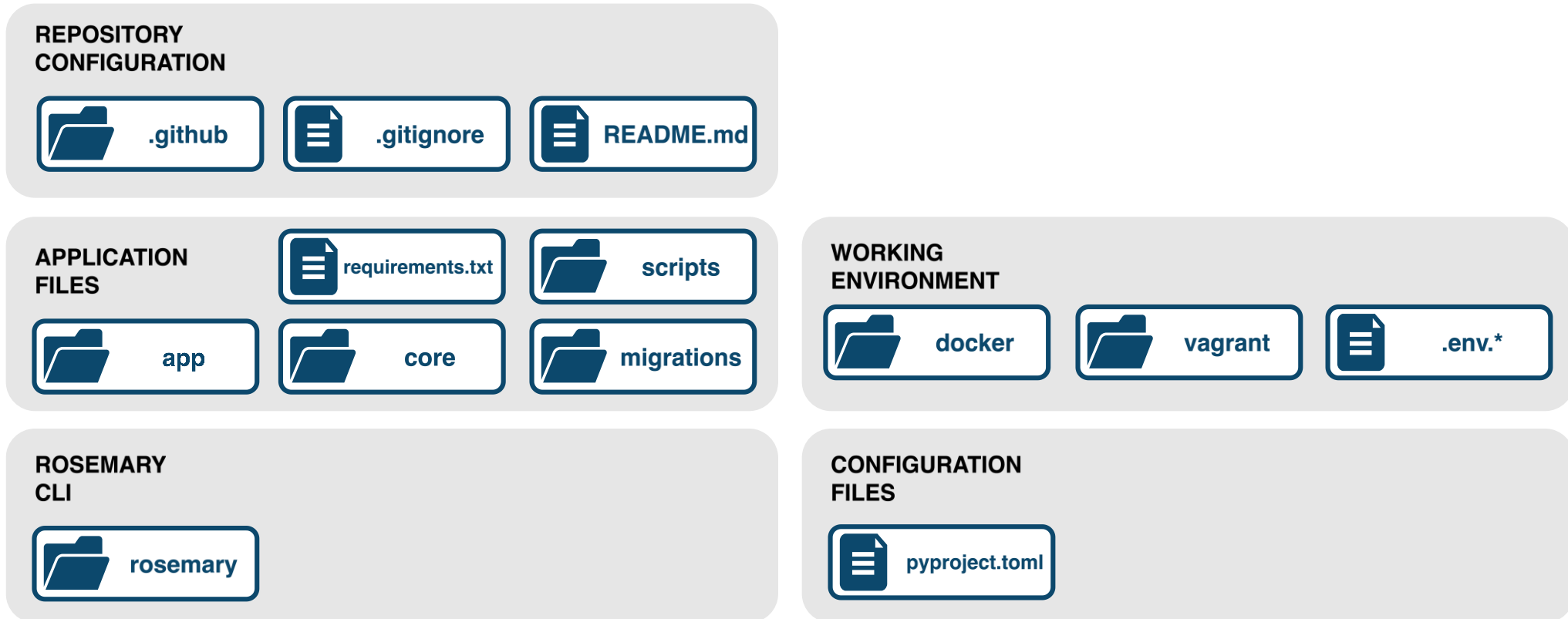
Petición HTTP



docs.uvlhub.io/architecture/http_request

3. Arquitectura de la aplicación

Estructura del proyecto



docs.uvlhub.io/architecture/project_structure

3. Arquitectura de la aplicación

Rosemary CLI



Rosemary es una herramienta CLI (*Command Line Interface*) pensada para facilitar la gestión de proyectos y las tareas de desarrollo

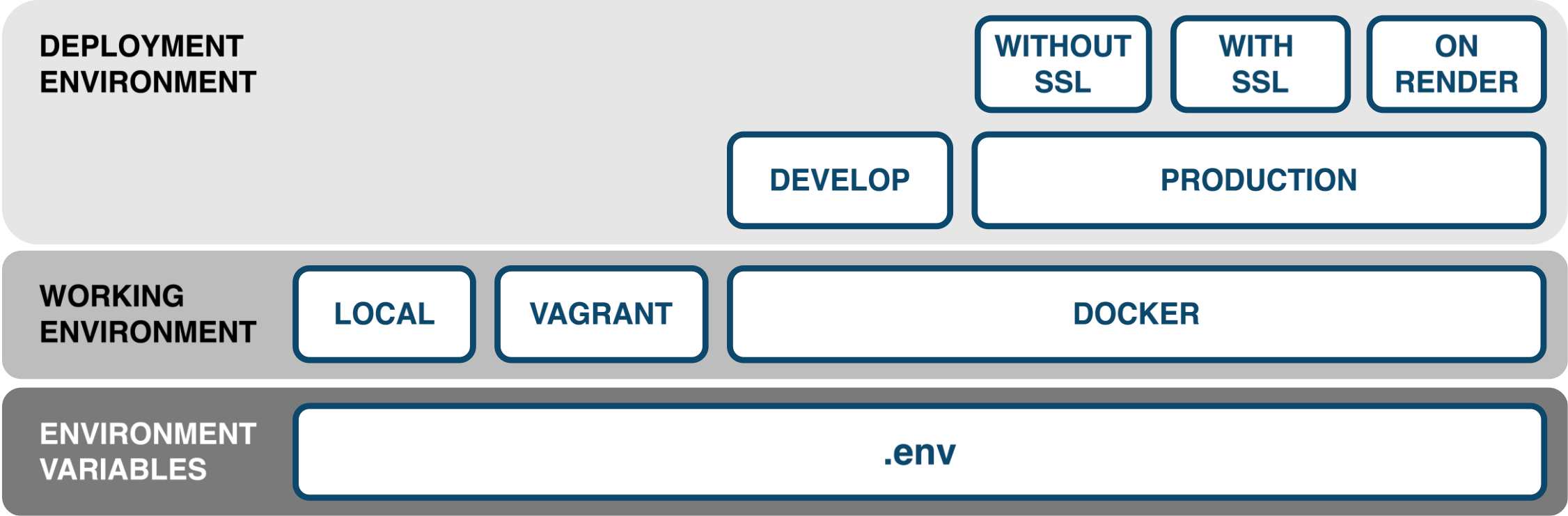


docs.uvlhub.io/rosemary

1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 - 4. Entornos**
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

4. Entornos

Entorno de despliegue VS. entorno de trabajo



4. Entornos

Variables de entorno

```
FLASK_APP_NAME="UVLHUB.IO(dev)"  
FLASK_ENV=development  
DOMAIN=localhost:5000  
MARIADB_PORT=3306  
MARIADB_HOSTNAME=localhost  
MARIADB_DATABASE=uvlhubdb  
MARIADB_TEST_DATABASE=uvlhubdb_test  
MARIADB_USER=uvlhubdb_user  
MARIADB_PASSWORD=uvlhubdb_password  
MARIADB_ROOT_PASSWORD=uvlhubdb_root_password  
WORKING_DIR=""
```

Archivo *.env* (local)

```
FLASK_APP_NAME="UVLHUB.IO(dev)"  
FLASK_ENV=development  
DOMAIN=localhost  
MARIADB_PORT=3306  
MARIADB_HOSTNAME=db  
MARIADB_DATABASE=uvlhubdb  
MARIADB_TEST_DATABASE=uvlhubdb_test  
MARIADB_USER=uvlhubdb_user  
MARIADB_PASSWORD=uvlhubdb_password  
MARIADB_ROOT_PASSWORD=uvlhubdb_root_password  
WORKING_DIR=/app/
```

Archivo *.env* (docker)

4. Entornos

Variables de entorno

```
FLASK_APP_NAME="UVLHUB.IO(dev)"  
FLASK_ENV=development  
DOMAIN=localhost:5000  
MARIADB_PORT=3306  
MARIADB_HOSTNAME=localhost  
MARIADB_DATABASE=uvlhubdb  
MARIADB_TEST_DATABASE=uvlhubdb_test  
MARIADB_USER=uvlhubdb_user  
MARIADB_PASSWORD=uvlhubdb_password  
MARIADB_ROOT_PASSWORD=uvlhubdb_root_password  
WORKING_DIR=""
```

Archivo *.env* (local)

```
FLASK_APP_NAME="UVLHUB.IO(dev)"  
FLASK_ENV=development  
DOMAIN=localhost  
MARIADB_PORT=3306  
MARIADB_HOSTNAME=db  
MARIADB_DATABASE=uvlhubdb  
MARIADB_TEST_DATABASE=uvlhubdb_test  
MARIADB_USER=uvlhubdb_user  
MARIADB_PASSWORD=uvlhubdb_password  
MARIADB_ROOT_PASSWORD=uvlhubdb_root_password  
WORKING_DIR=/app/
```

Archivo *.env* (docker)

1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

5. ¿Qué necesitamos?

REQUISITOS **HARDWARE**



- **Intel Core i5 de 10ª** generación o **AMD Ryzen 5** de la serie 4000
- **8 GB** de RAM DDR4
- SSD o NVMe de **256 GB**
- **100 GB de espacio en disco** (partición en Ubuntu o MV)

REQUISITOS **SOFTWARE**



- **Ubuntu 22.04 LTS** o superior
- **Python 3.12**
- Docker, Docker compose
- Virtualbox, Vagrant, Ansible
- Visual Studio Code
- Git

5. ¿Qué necesitamos?

Máquina Vagrant oficial de la asignatura



- **Imagen oficial** de los ordenadores de las aulas de prácticas
- Ideal para practicar en casa con el mismo entorno que el examen
- Sin embargo, **no permite virtualización anidada**

https://portal.cloud.hashicorp.com/vagrant/discover/cdcetsii/UbuntuEGC_25

5. ¿Qué necesitamos?

Máquina Vagrant oficial de la asignatura

Crear carpeta con el siguiente **Vagrantfile**



```
Vagrant.configure("2") do |config|  
  
  config.vm.box = "cdketsii/UbuntuEGC_25"  
  
  config.vm.provider "virtualbox" do |vb|  
    vb.name = "UbuntuEGC"  
    vb.memory = "2048"  
    vb.cpus = 2  
  end  
  
  config.vm.network "forwarded_port", guest: 5000, host: 5000  
  
end
```

5. ¿Qué necesitamos?

Máquina Vagrant oficial de la asignatura

Crear y levantar máquina

```
vagrant up
```

Acceder a la máquina

```
vagrant ssh
```

Parar la máquina

```
vagrant halt
```

5. ¿Qué necesitamos?

Nuestro IDE: Visual Studio Code



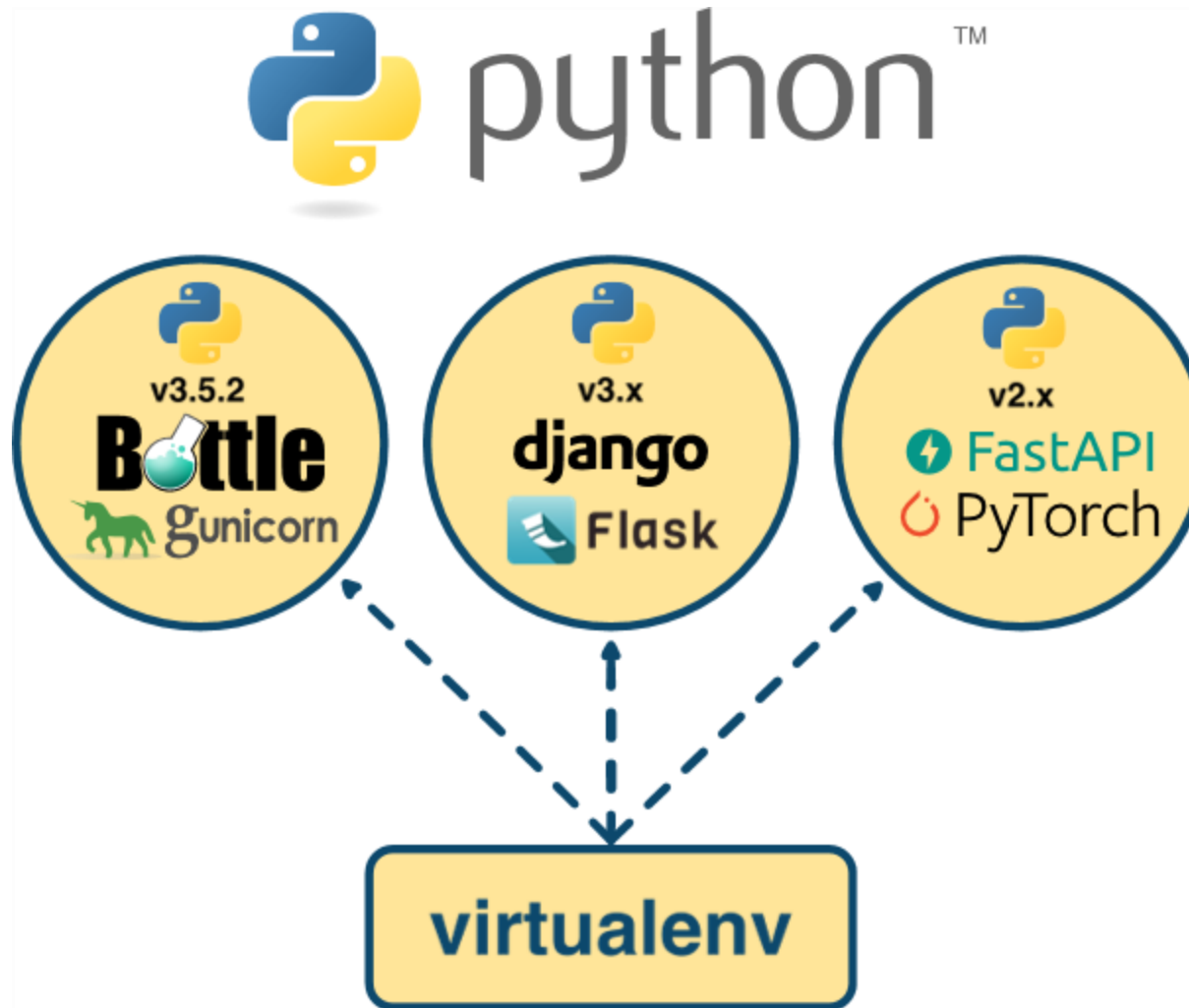
EXTENSIONES ÚTILES

- Docker
- Python
- Debugpy
- Python indent
- Flake8
- ESLint
- PyPi Assistant
- Flask Snippets

1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. **Python: entornos virtuales**
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

6. Python: entornos virtuales

¿Qué son?



6. Python: entornos virtuales

Comandos básicos

Crear entorno virtual

```
python3.12 -m venv <myenvname>
```

Activar entorno virtual

```
source <myenvname>/bin/activate
```

Desactivar entorno virtual

```
deactivate
```

1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 - 7. Ejercicio práctico: configurar GitHub y forkear**
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

7. Ejercicio práctico: configurar GitHub y forkear

Git: instalar y configurar

```
sudo apt install git  
git config --global user.name "<tu nombre y apellidos entrecomillados>"  
git config --global user.email <tu email>
```

Git: generar par de claves SSH

```
ssh-keygen -t rsa -b 4096
```

7. Ejercicio práctico: configurar GitHub y forkear

Git: visualizar clave pública y copiar

```
cd ~/.ssh  
cat id_rsa.pub
```

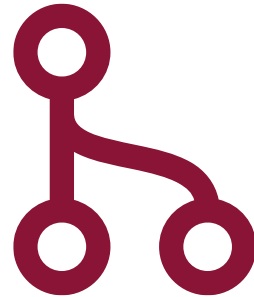
Git: guardar clave pública en GitHub

GitHub -> Settings -> SSH and GPG keys -> New SSH key -> Pegar clave pública (id_rsa.pub)

7. Ejercicio práctico: configurar GitHub y forkear



github.com/EGCETSII/uvlhub



FORK!



github.com/<tuusuario>/uvlhub_practicas



1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 - 8. Ejercicio práctico: instalar uvlhub**
 9. Ejercicio práctico: tutorial C.R.U.D.
 10. Anexo: migraciones en colaborativo
- 

8. Ejercicio práctico: instalar uvlhub



docs.uvlhub.io/installation/manual_installation

Docs / Installation / Manual installation

1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 - 9. Ejercicio práctico: tutorial C.R.U.D.**
 10. Anexo: migraciones en colaborativo
- 

9. Ejercicio práctico: tutorial C.R.U.D.



docs.uvlhub.io/tutorials/crud_tutorial



Docs / Tutorials / C.R.U.D. tutorial

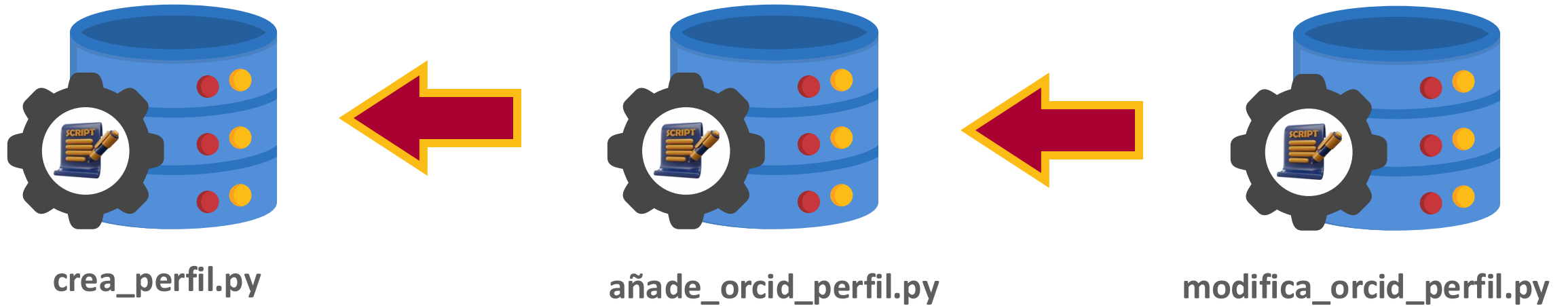
1. ¿Qué es uvlhub.io?
 2. Documentación oficial
 3. Arquitectura de la aplicación
 4. Entornos
 5. ¿Qué necesitamos?
 6. Python: entornos virtuales
 7. Ejercicio práctico: configurar GitHub y forkear
 8. Ejercicio práctico: instalar uvlhub
 9. Ejercicio práctico: tutorial C.R.U.D.
 - 10. Anexo: migraciones en colaborativo**
- 

10. Anexo: migraciones en colaborativo

CONCEPTO DE MIGRACIÓN

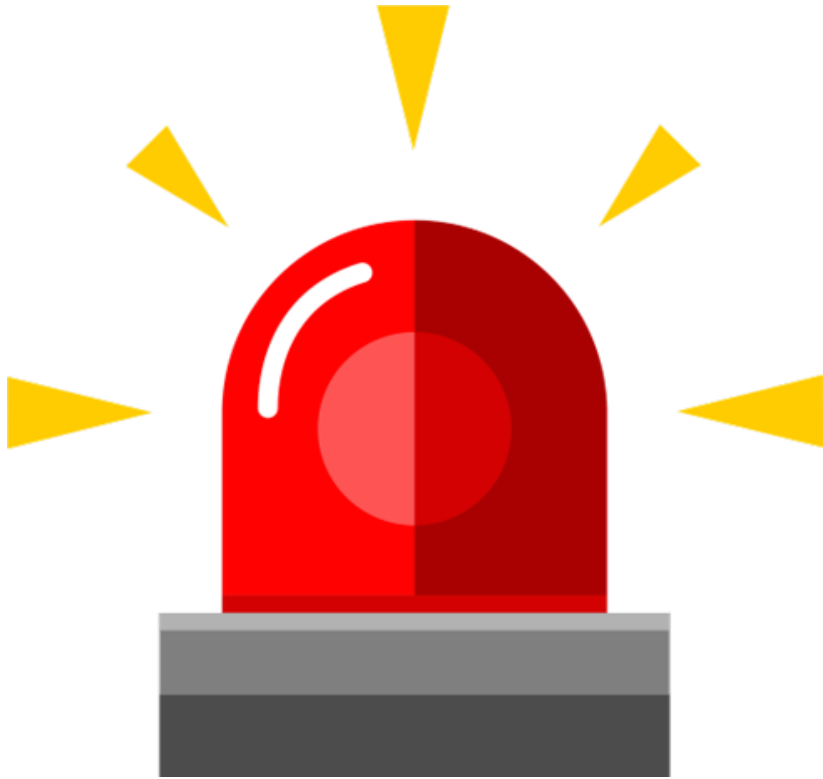
Mecanismo que permite gestionar la evolución del esquema de una base de datos mediante cambios incrementales y reversibles, con el fin de mantener la coherencia entre los datos y el código de la aplicación a lo largo del ciclo de vida del software.

10. Anexo: migraciones en colaborativo



Un archivo de migraciones siempre
apunta al anterior

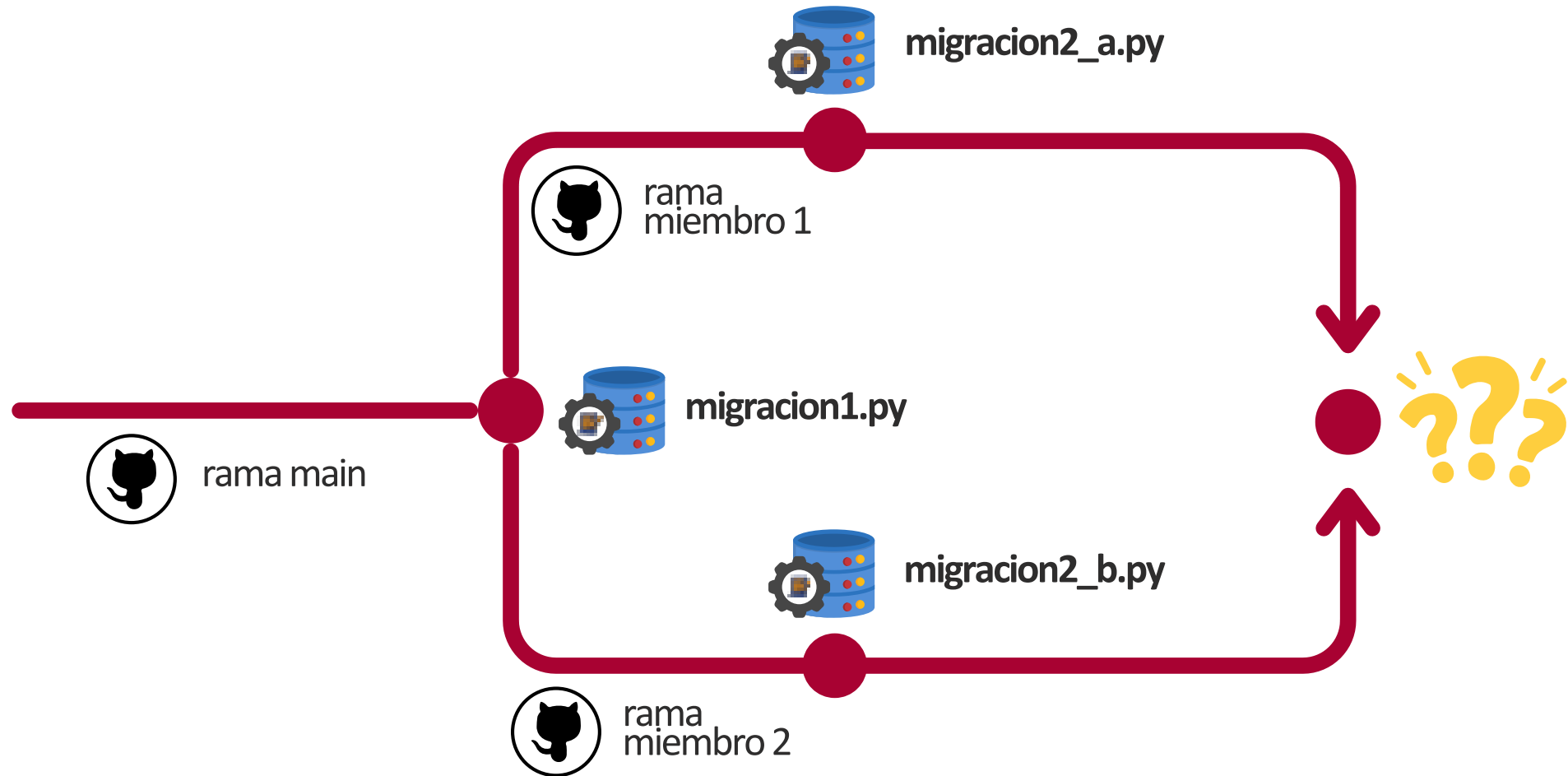
10. Anexo: migraciones en colaborativo



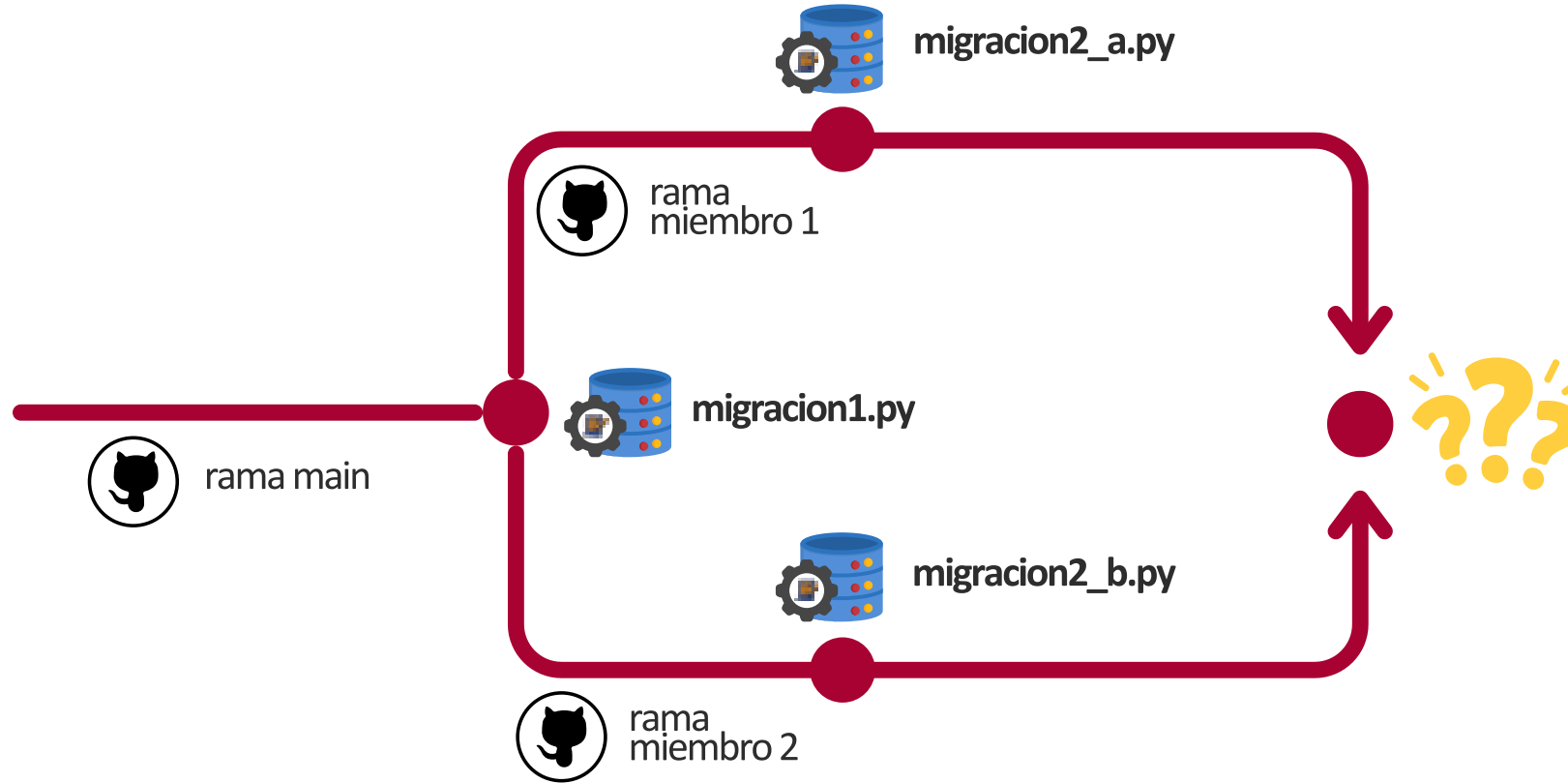
PROBLEMA

Dos miembros del equipo implementan cambios en el modelo y parten de la misma migración

10. Anexo: migraciones en colaborativo

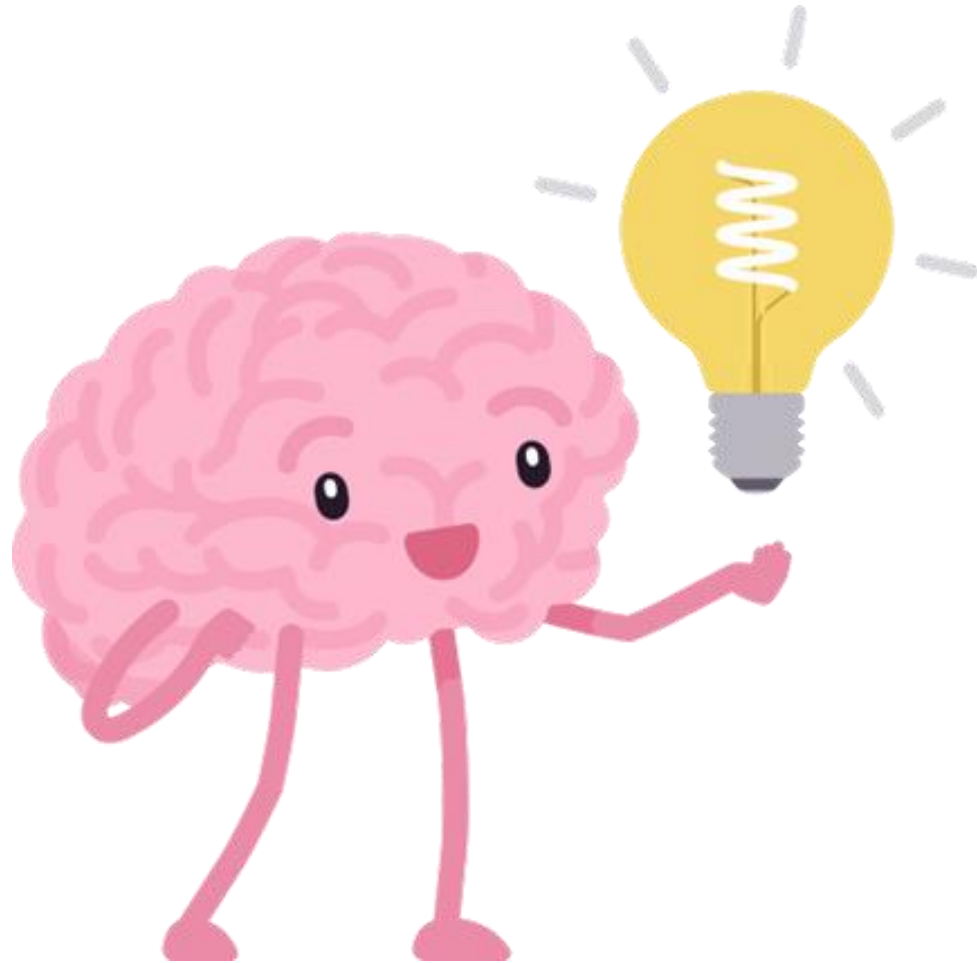


10. Anexo: migraciones en colaborativo



¡No podemos tener dos migraciones apuntando a la misma!

10. Anexo: migraciones en colaborativo



10. Anexo: migraciones en colaborativo

OPTION 1



OPTION 2



10. Anexo: migraciones en colaborativo



CONSEJO #1

Nombrad las migraciones en orden incremental, e.g., **001.py**, **002.py**, **003.py**... así es mucho más fácil saber qué apunta a qué

10. Anexo: migraciones en colaborativo



CONSEJO #2

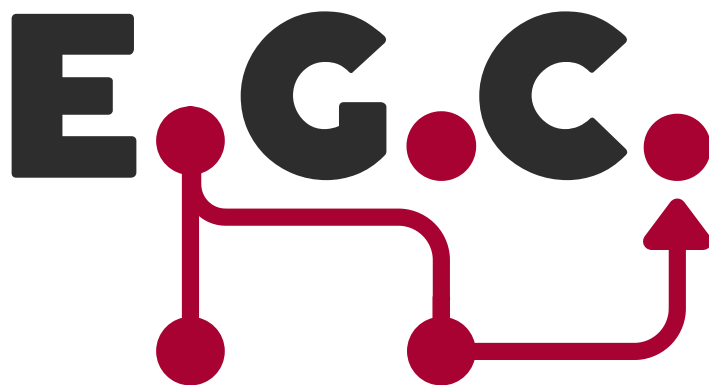
Si alguien antes que tú ya ha apuntado su migración a la migración base y está incorporado en la rama *main*, **TE TOCA A TI** solucionar el conflicto con tu migración

10. Anexo: migraciones en colaborativo



CONSEJO #3

Borrar todas las migraciones y resetear la base en cada conflicto **ES UN SINSENTIDO** y rompe el concepto en sí de migración -> **ievaluaremos negativamente!**



Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración



Escuela Técnica Superior de
Ingeniería Informática

¡Gracias!

“En una jerarquía, todo empleado tiende a ascender hasta su nivel más alto de incompetencia.”

- Lawrence J. Peter