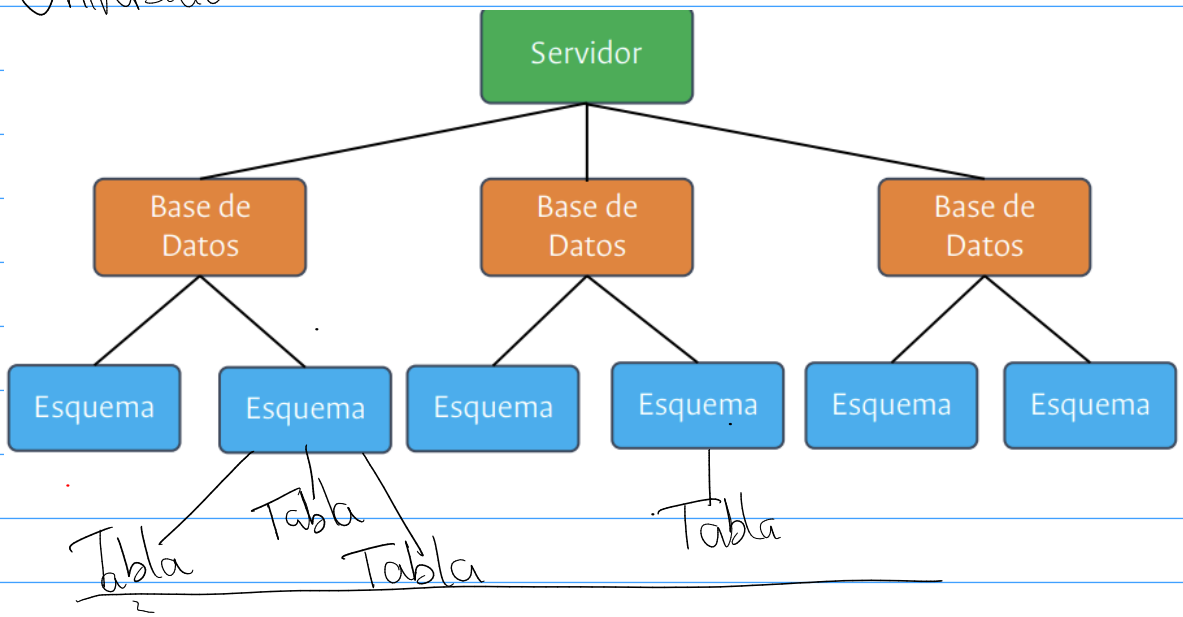
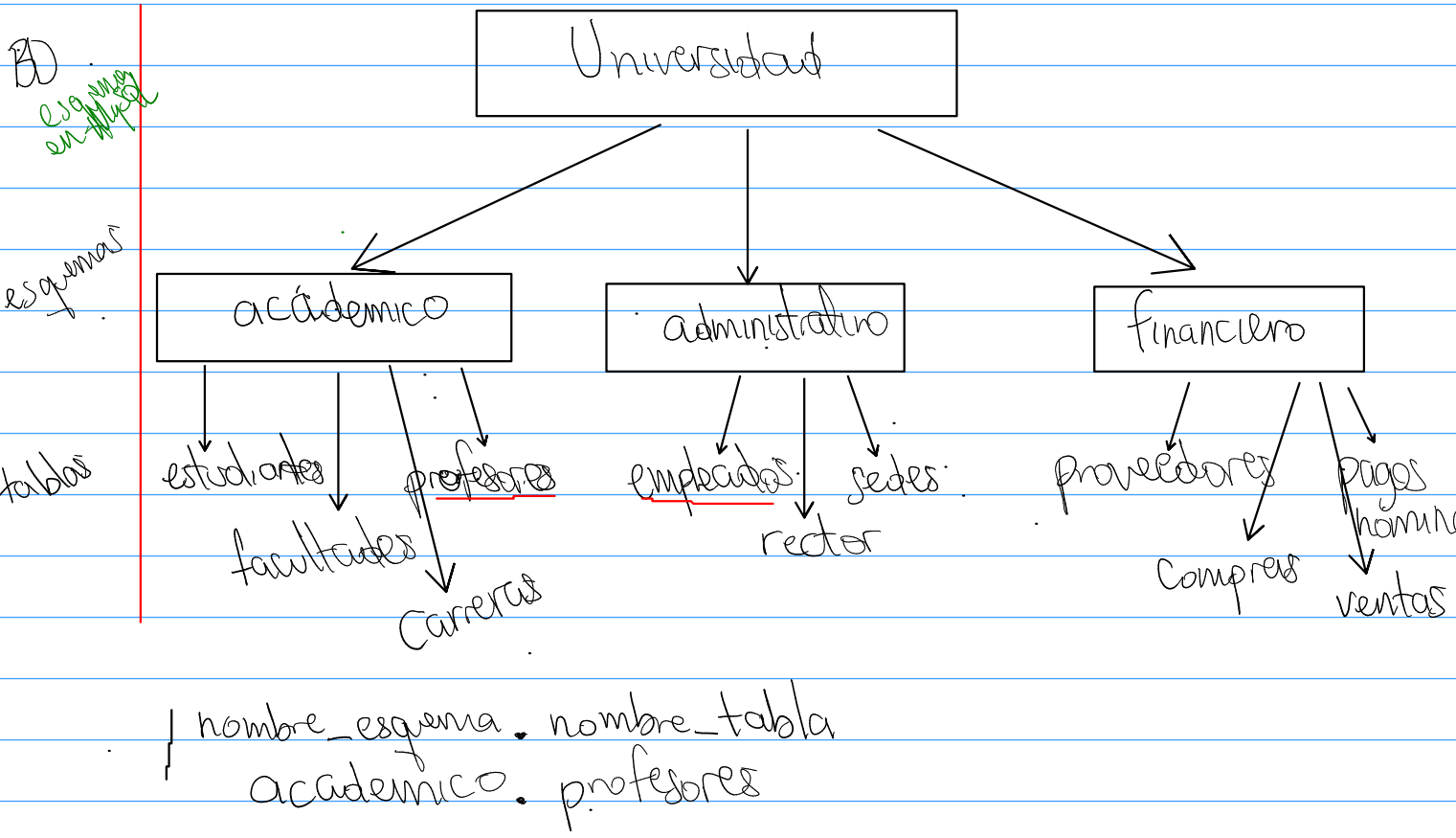
usuario pg
pass postgres
postgres

mysql
root
root

Universidad





nombre_esquema.nombre_tabla
academico.profesores

Aplicación para un cajero con arquitectura de 3 capas

1. Lógica → Backend
2. Presentación → Frontend
3. Datos → Base de datos (ficticia por ahora)

BD — Driver — Lógica — Presentación

aún no está hecho

Backend

- ✓ Python
- ✓ FastAPI
- ✓ pydantic (validación y tipado)
- ✓ apirest
- ✓ uvicorn
- ✓ typing
- ✓ Postman
- ✓ CORS
- ✓ Swagger

Frontend

- ✓ HTML
- ✓ Javascript (JS)
- ✓ CSS
- ✓ VueJS
- ✓ HTTP
- ✓ router
- ✓ babel
- ✓ node.JS
- ✓ axios
- ✓ express

Despliegue

- github — web donde alguien repos. — otros: gitlab, bitbucket
- git — sistema de control de versiones
- Heroku

GIT

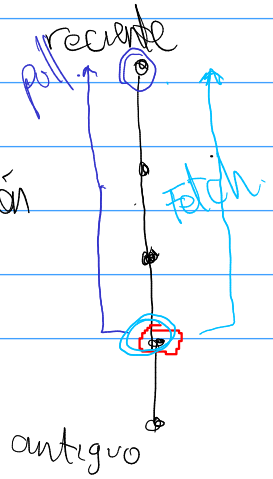
pull: Trae del repo en la nube. Hace actualización del ~~HEAD~~

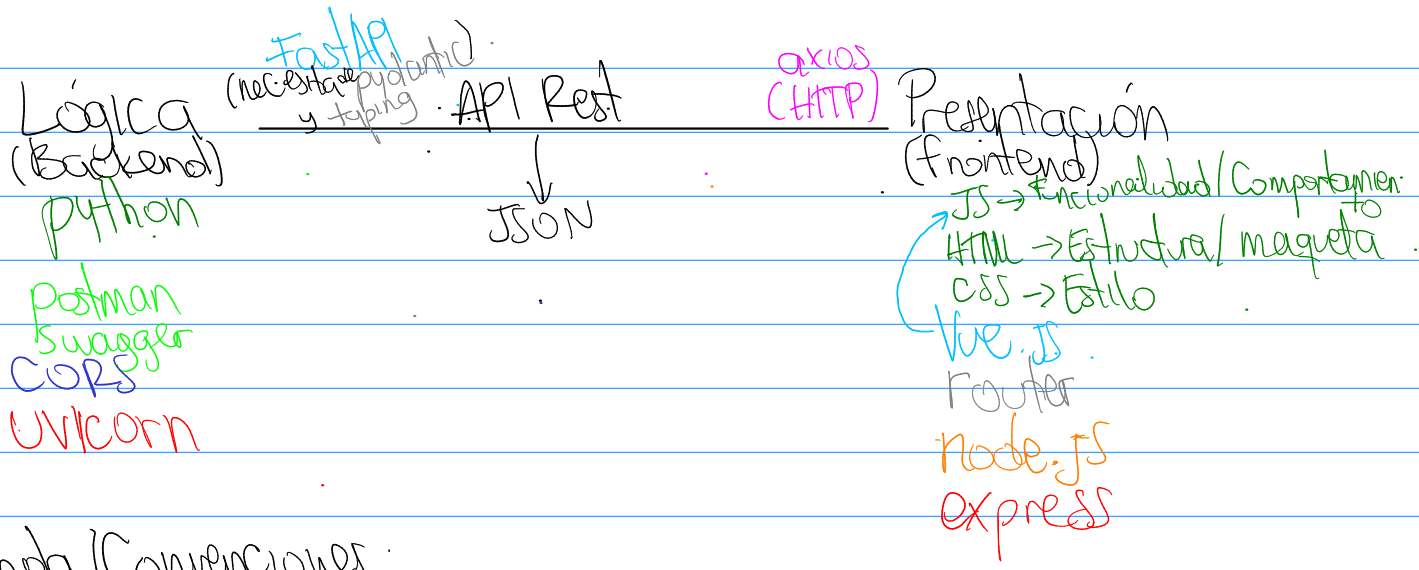
push: Lleva desde el local hacia la nube

branch (rama)

revert o rollback

fetch: Actualiza pero no hace actualización del HEAD





Leyenda/Convenciones:

- Lenguaje
- Frameworks: Para reutilizar. Nos brinda funcionalidades que nos acortarán el camino. Permiten llegar muy lejos con menos trabajo
- Librería
- Servidor: El que está pendiente de las peticiones que le haga el cliente. Queda funcionando como un demonio.
- Herramientas adicionales
- Cliente
- Manejador de dependencias

CORS (Políticas): Es el backend diciendo a quien quiere darle permisos para que lo utilicen.

Postman: Es un programa que permite verificar/probar las APIs que tenemos disponibles

Swagger: Documentación de las apis. FastAPI lo utiliza para crear la documentación de apis automáticamente.

router: Para definir URLs. Para enrutar

babel: Para tener en cuenta a las versiones antiguas dar soporte

node.js: En este proyecto lo estamos utilizando para
↓ · manejar las dependencias del frontend. Está ayudando
npm a que alistemos/gestionemos nuestro frontend.