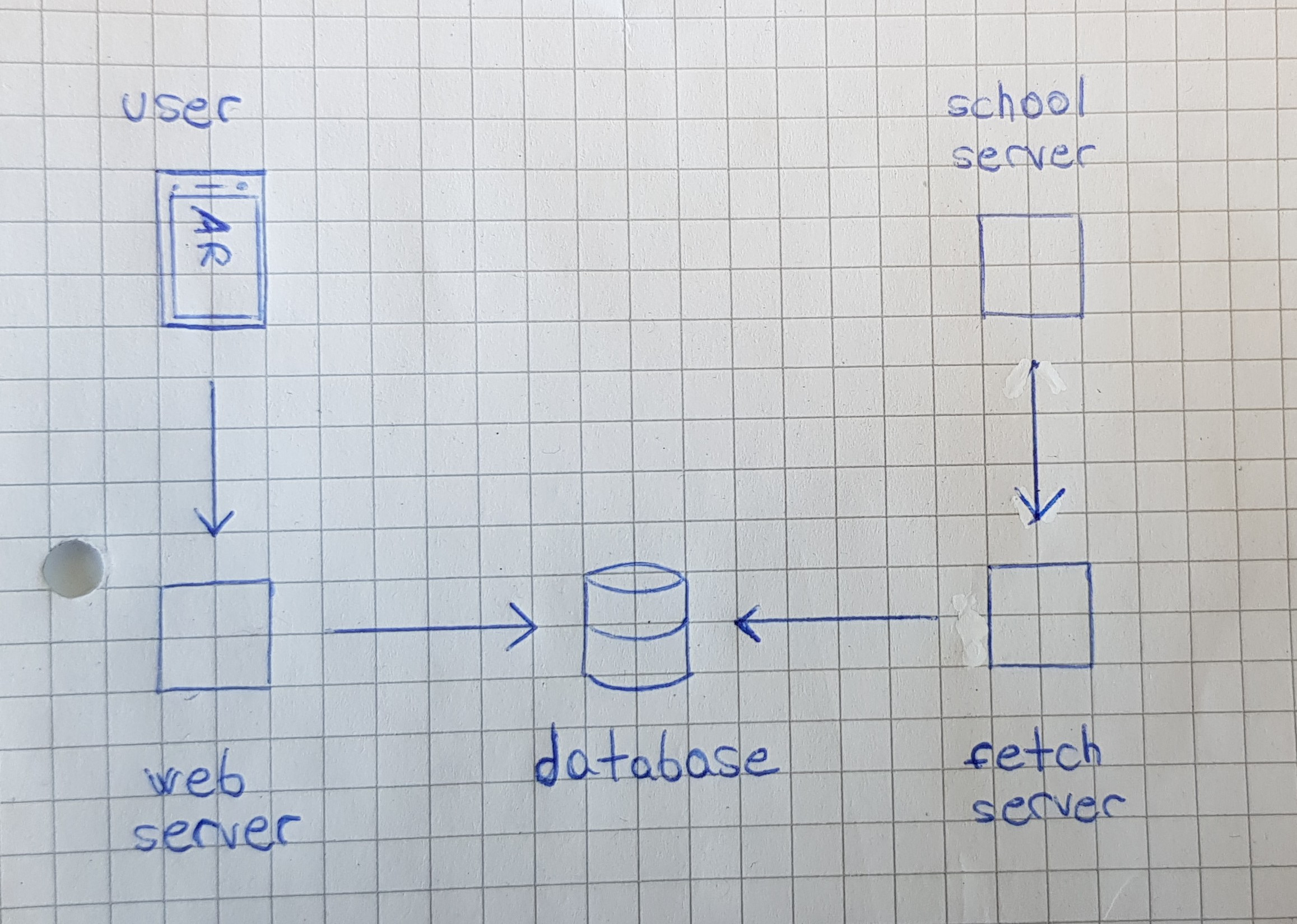
**Diario di lavoro**

|  |  |
| --- | --- |
| Luogo | Scuola d'Arti e Mestieri di Trevano, Canobbio |
| Data | 07 Febbraio 2020 |

Lavori svolti

Lorenzo: ho terminato il prototipo per ar.js. Ho in seguito iniziato a progettare all'architettura del server (web server, fetching degli orari, ...) e a sviluppare lo scheletro del server. Usiamo Docker e docker-compose per facilitare l'installazione del server, Python come linguaggio server-side e Flask come web framework.

(sopra: schema server)

Web server: il server che hosta il sito con il quale gli utenti interagiscono. Esso recupera i dati degli orari del database.

Database: gli orari di tutte le aule.

Fetch server: server che estrae gli orari del server scolastico e li inserisce del database. è ancora da stabilire quando il server scaricherà i dati.

Il tutto non è ancora definitivo.

Julian: ho finito il sito con l'auto di Bootstrap, come consigliato dal sore.

Problemi riscontrati e soluzioni adottate

Lorenzo: ho incontrato molti problemi tentando di generare dei markers dinamicamente da uno script JavaScript; non ho risolto il problema, ma ho deciso di generarli direttamente sul server e non sul client.

Julian: ho riscontrato varie difficoltà con CSS ma risolte con Bootstrap.

Pierpaolo e Claudio: il sito della scuola è offuscato, rendendo quasi impossibile l'estrazione dei dati. Il sore ci ha passato un pezzo di codice utile per estrarre i dati. Il programma si chiama Scrapper ed avvia i test di Selenium.

Codice per far funzionare Scrapper:

* self.browser.get(self.baseurl): permette di fare un GET sul sito
* cerca\_orario = self.browser.find\_element\_by\_id("id"): cercare un elemento con un determinato ID
* cerca\_orario.click(): permette di cliccare un elemento
* cerca\_orario.send\_keys("ciao"): permette di scrivere una stringa all'interno di un TextBox.
* Abbiamo incontrato un problema quando la pagina web non rispondeva all'input dalla tastiera. Abbiamo risolto il problema mandando l'input al sistema operativo invece che solo al browser.
* Preme i tasti giù e invio per cambire la selezione da griglia ad elenco

from pynput.keyboard import Key, Controller: Liberia da importare per utilizzare keyboard del sistema operativo

Utilizziamo un time.sleep perché quando clicchiamo il menu a tendina per scegliere la visualizzazione c'è un'animazione che crea ritardi.

Ricordare di importare libreria time => import time

k = Controller()

time.sleep(1)

k.press(Key.down)

time.sleep(1)

k.release(Key.down)

k.press(Key.enter)

k.release(Key.enter)

time.sleep(2)

* Problema: Fare i parse del HTML in generale

Soluzione: Utilizzare la libreria Beautiful soup (Doc: https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

Utilizzare espressioni regolari per filtrare i dati (https://regex101.com/)

soup = BeautifulSoup(self.browser.page\_source): permeate di ricavare tutto l'HTML

table = soup.find('table', id="GInterface.Instances[1].Instances[8]\_Contenu\_0"): cerca la tabella che contiene l'orario

rows = table.find\_all\_next('tr', {"class": "AvecMain c\_7"}): cerca le righe che contengono le info sulle materie (orario inizio e fine, docente, ...)

giorni = table.find\_all\_next('td', {"class": "Gras"}): cerca nella tabella il giorno (elementi td che hanno un classe di nome Gras)

for i, row in enumerate(rows): cicla su ogni riga

for el in row.find\_all\_next('td'): ciclo su ogni td contenuto in ogni tr

print(el.text)

giorno = ""

for g in giorni: ciclo su ogni giorno

pattern = r'(?P<giorno>\w{3}).\*\s(?P<dd>\d\d)\s(?P<mese>\w{3,10})\s+(?P<anno>\d\d\d\d)'

m = re.search(pattern, g.text): Es: lunedì 03 febbraio 2020 => Gruppo giorno: lun , Gruppo dd: 03 , Gruppo mese: febbraio, Gruppo anno: 2020

if m:

giorno = m.group('giorno')

Punto della situazione rispetto alla pianificazione

Siamo nei tempi giusti essendo che abbiamo già iniziato il prototipo.

Programma di massima per la prossima giornata di lavoro

Lorenzo: finalizzare la struttura del server e iniziarne lo sviluppo.

Pierpaolo + Claudio: terminare il parser dei dati e salvari i dati in formato JSON.