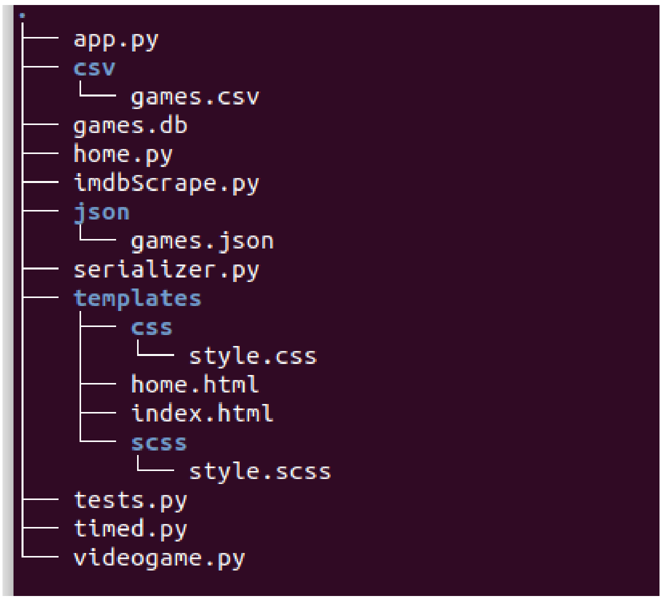
Detail report

Python project 2018

Rabeb Jaouadi

EPITA
12 July 2019

Architecture :



```
.
├── app.py
├── csv
│   └── games.csv
├── games.db
├── home.py
├── imdbScrape.py
├── json
│   └── games.json
├── serializer.py
├── templates
│   ├── css
│   │   └── style.css
│   ├── home.html
│   ├── index.html
│   └── scss
│       └── style.scss
├── tests.py
├── timed.py
└── videogame.py
```

For our program we define the following classes :

- App : Initializes the app of the

- videogame : Datamodel with all the info that

- home.py : main method of the app with CRUD functions

- imdbScrape : all scrapping methods

- template :

– index.html : home page of the app

– style.css : style of the home page

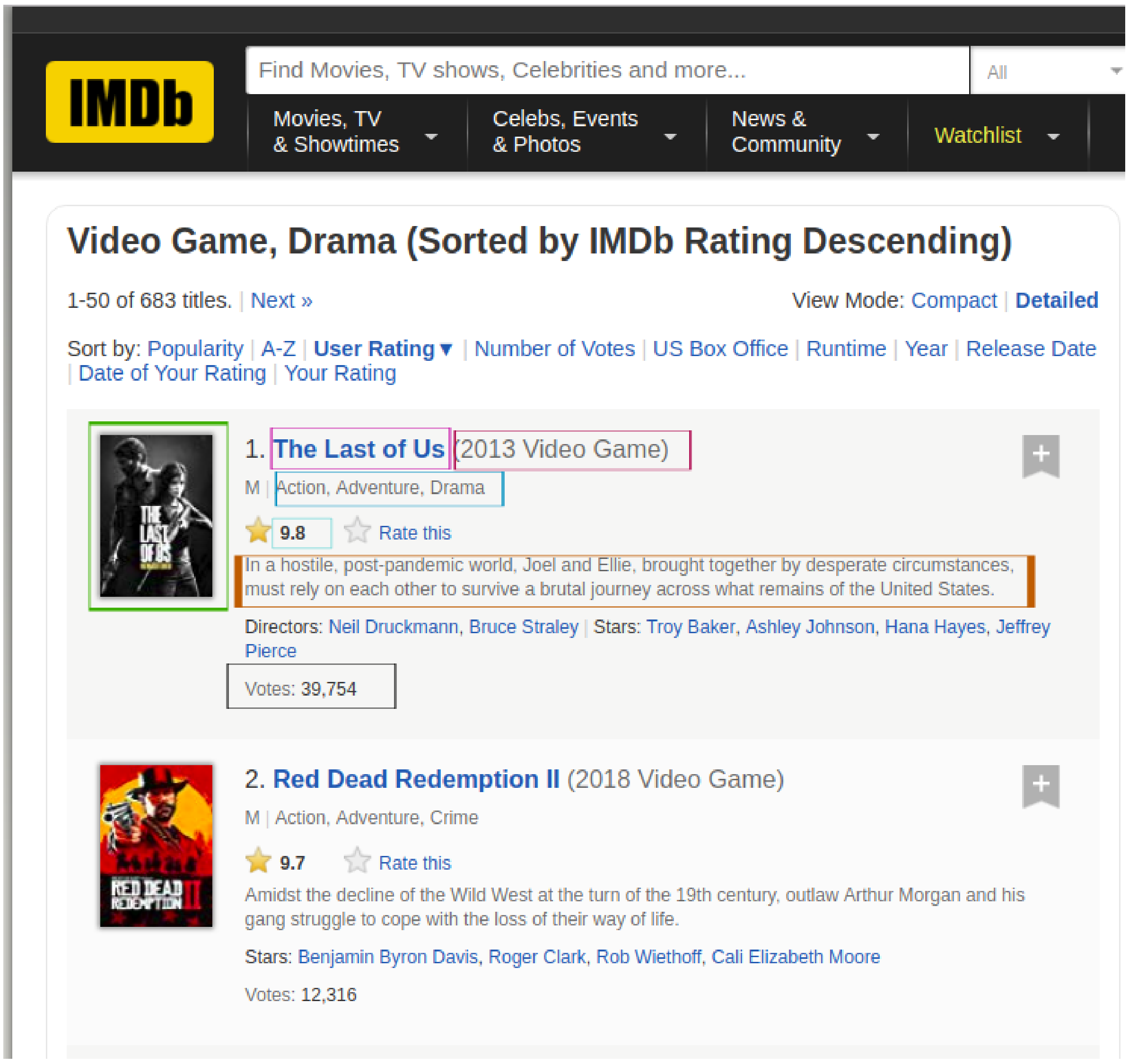
-tests : a set of tests to test the scrapping methods

- serializer : serializes to csv or json

- timed : Decorator to log the name of the function and the time it's been running for

Dataset :

The dataset chosen to scrape is IMDB video games sorted by rating



Video Game, Drama (Sorted by IMDb Rating Descending)

1-50 of 683 titles. | Next »

View Mode: Compact | Detailed

Sort by: Popularity | A-Z | User Rating ▾ | Number of Votes | US Box Office | Runtime | Year | Release Date | Date of Your Rating | Your Rating

1. The Last of Us (2013 Video Game)

M | Action, Adventure, Drama

9.8 Rate this

In a hostile, post-pandemic world, Joel and Ellie, brought together by desperate circumstances, must rely on each other to survive a brutal journey across what remains of the United States.

Directors: Neil Druckmann, Bruce Straley | Stars: Troy Baker, Ashley Johnson, Hana Hayes, Jeffrey Pierce

Votes: 39,754

2. Red Dead Redemption II (2018 Video Game)

M | Action, Adventure, Crime

9.7 Rate this

Amidst the decline of the Wild West at the turn of the 19th century, outlaw Arthur Morgan and his gang struggle to cope with the loss of their way of life.

Stars: Benjamin Byron Davis, Roger Clark, Rob Wiethoff, Cali Elizabeth Moore

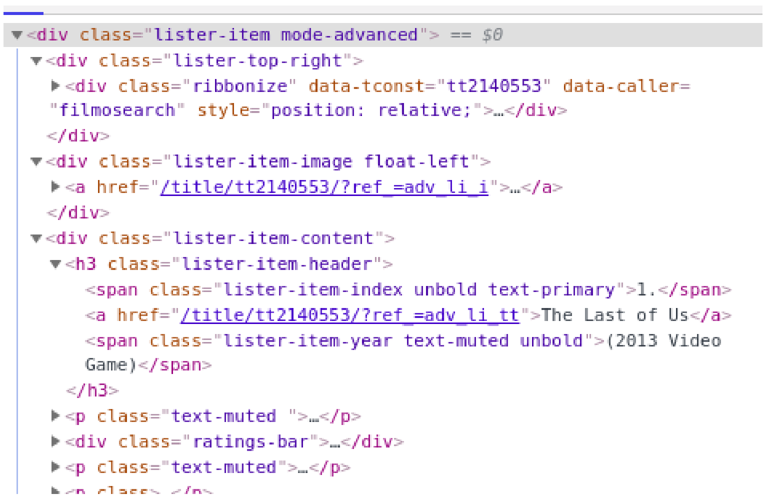
Votes: 12,316

We scrape the following data :

- Game's title
- Link to the game
- Image of the game
- Genre
- Release date
- IMDB rating
- Description

- Number of votes

Scrapers :



```
▼<div class="lister-item mode-advanced"> == $0
  ▼<div class="lister-top-right">
    ▶<div class="ribbonize" data-tconst="tt2140553" data-caller=
    "filmosearch" style="position: relative;">…</div>
    </div>
  ▼<div class="lister-item-image float-left">
    ▶<a href="/title/tt2140553/?ref_=adv_li_i">…</a>
    </div>
  ▼<div class="lister-item-content">
    ▼<h3 class="lister-item-header">
        <span class="lister-item-index unbold text-primary">1.</span>
        <a href="/title/tt2140553/?ref_=adv_li_tt">The Last of Us</a>
        <span class="lister-item-year text-muted unbold">(2013 Video
        Game)</span>
      </h3>
    ▶<p class="text-muted ">…</p>
    ▶<div class="ratings-bar">…</div>
    ▶<p class="text-muted">…</p>
```

We define the following functions :

- Scrape all games : findall elements with the tag "lister-item mode-advanced" which corresponds to a bloc of info concerning one game. we search in that all element that all the data we are interested in : e.g :

```
            for element in bs.findAll("div",
                    {"class": "lister-item mode-advanced"})[:data_limit]:
                game = Videogame()
                game.set_link("http://www.imdb.com" + element.find("a")['href'])
```

- Scrape poster by title : comparing the title given with the title scraped from all elements if one title matches, scrape the poster from the element

```
        if title.lower() in game_title.lower():
            poster = element.find("img")['loadlate']
```

- Scrape full poster by title : same as previous function compare title if matches find the link of the game and find the poster in the convenient element in the links page

```
        if title.lower() in game_title.lower():
            link = "http://www.imdb.com" + element.find("a")['href']
            game_html = urlopen(link)
            bs = BeautifulSoup(game_html, 'html.parser')
            poster = "http://www.imdb.com" + bs.find("div", {"class": "poster"}).a['
                                                      href']
```

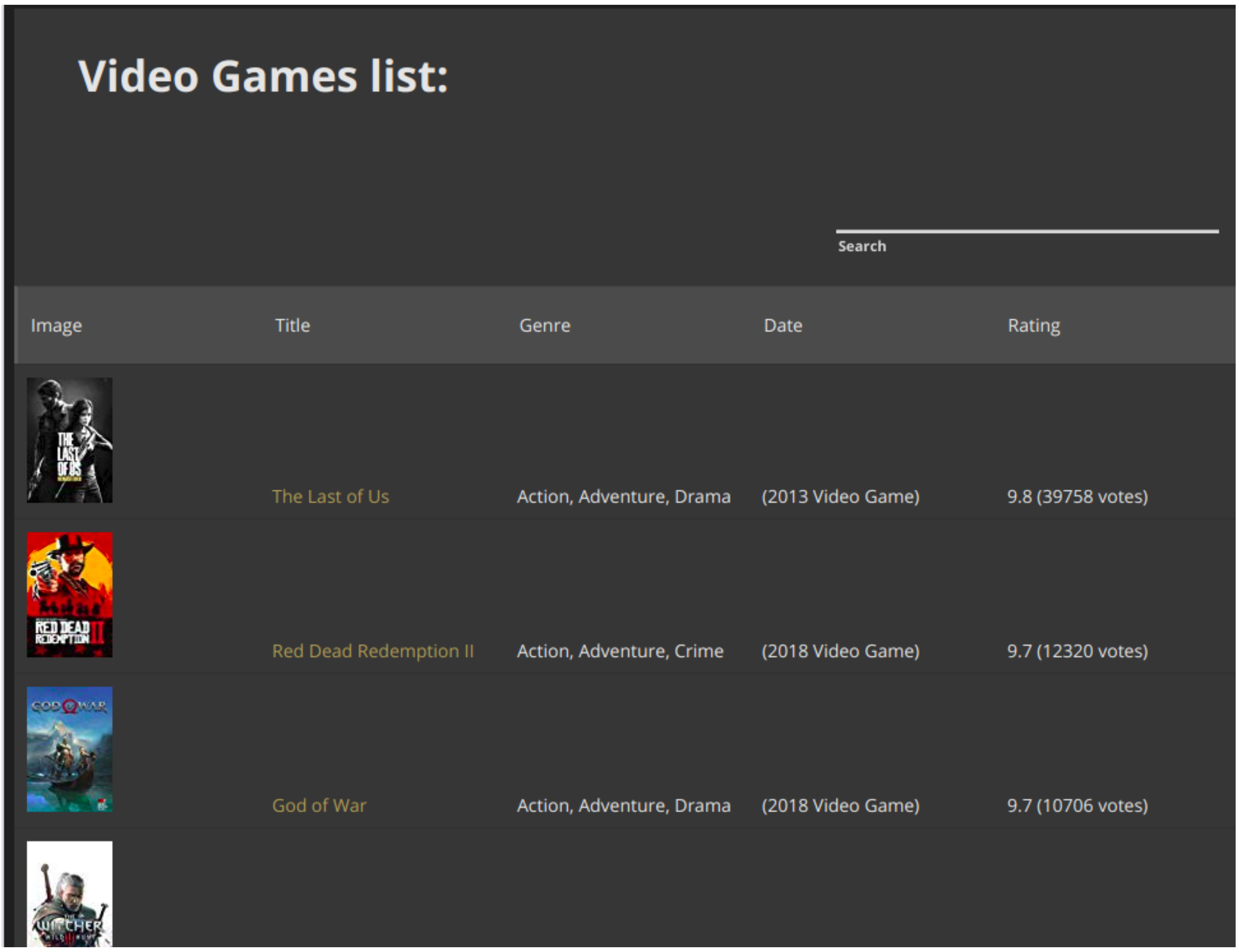
-Scrape all posters of all games

- Display all games using toString() defined in the class videogame.

Front-end Back-end :

We define the following functions :

- Display all games : when we open the main page a list of all games is displayed. Clicking on the title of a game will redirect to the imdb page of the game. Hovering over the game will unscroll a screen that displays the description of the game.



Video Games list:

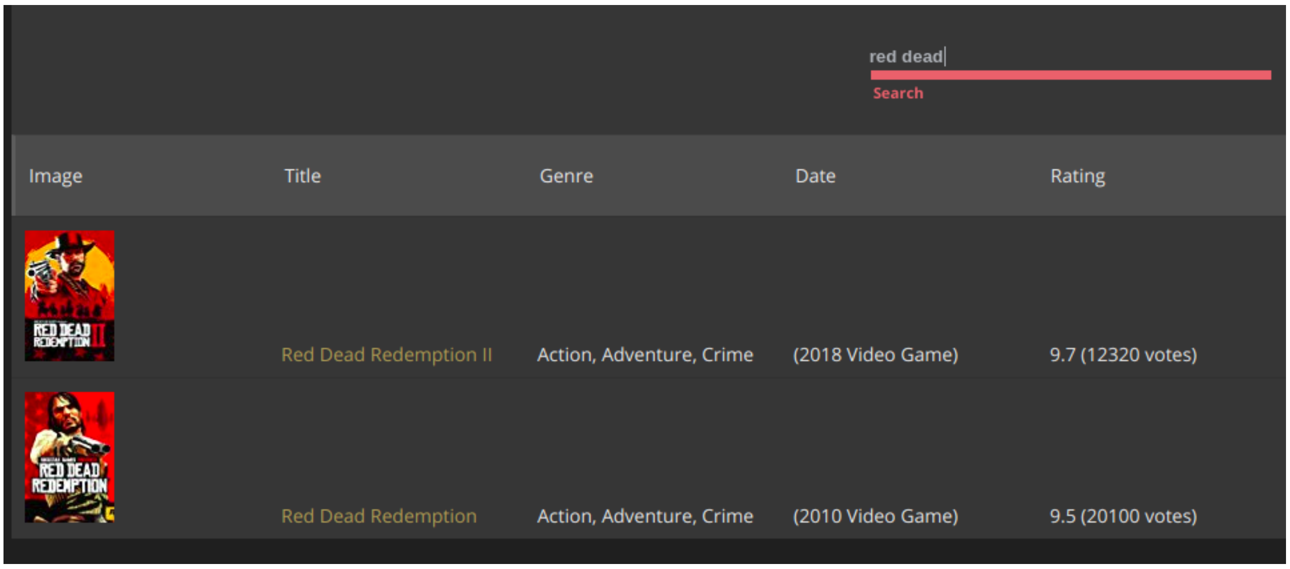
Search

Image	Title	Genre	Date	Rating
	The Last of Us	Action, Adventure, Drama	(2013 Video Game)	9.8 (39758 votes)
	Red Dead Redemption II	Action, Adventure, Crime	(2018 Video Game)	9.7 (12320 votes)
	God of War	Action, Adventure, Drama	(2018 Video Game)	9.7 (10706 votes)

code :

```
@timed(enabled=True)
@app.route("/", methods=["GET", "POST"])
def home():
    games = Videogame.query.all()
    return render_template("index.html", games = games)
```

- Search a game : typing a title of a game, or part of the title will display all the games matching



red dead

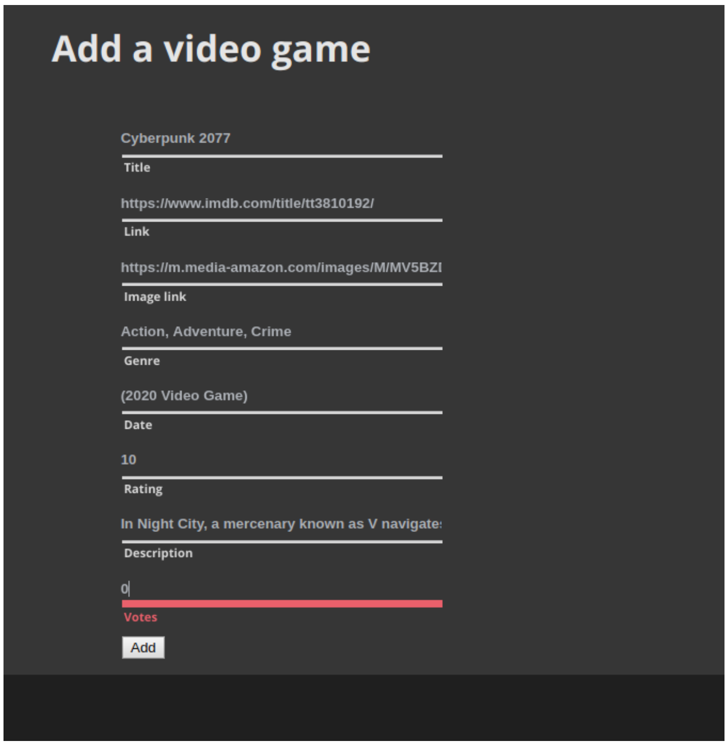
Search

Image	Title	Genre	Date	Rating
	Red Dead Redemption II	Action, Adventure, Crime	(2018 Video Game)	9.7 (12320 votes)
	Red Dead Redemption	Action, Adventure, Crime	(2010 Video Game)	9.5 (20100 votes)

code :

```
    def search():
        search_title = request.form.get("search_title")
        found_games = Videogame.query.filter(
                Videogame.title.contains(search_title)).all()
        return render_template("index.html", games = found_games)
```

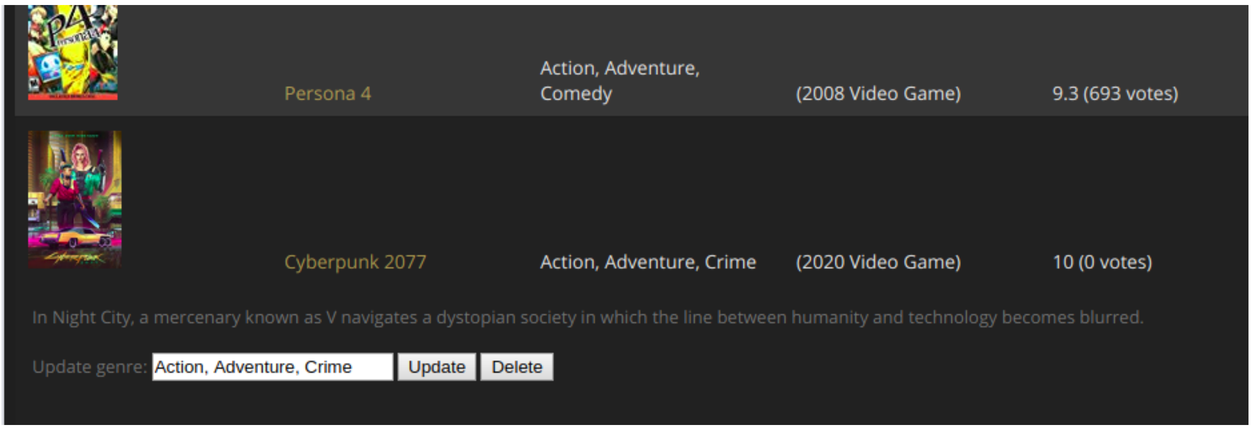
- Adding a game : In order to add a game the user has to fill the form on top of the page with the game information



Add a video game

Title	Cyberpunk 2077
Link	https://www.imdb.com/title/tt3810192/
Image link	https://m.media-amazon.com/images/M/MV5BZI
Genre	Action, Adventure, Crime
Date	(2020 Video Game)
Rating	10
Description	In Night City, a mercenary known as V navigate:
Votes	0

The game will then be added in the database and in the display of the page



	Persona 4	Action, Adventure, Comedy	(2008 Video Game)	9.3 (693 votes)
	Cyberpunk 2077	Action, Adventure, Crime	(2020 Video Game)	10 (0 votes)

In Night City, a mercenary known as V navigates a dystopian society in which the line between humanity and technology becomes blurred.

Update genre: Action, Adventure, Crime Update Delete

Clicking on delete in front of any game will delete the game in question. To update the user has to change the genre of the game and hit update. the new genre will be added in the database replacing the old one.

e.g : Delete code

```
    @timed(enabled=True)
    @app.route("/delete", methods=["POST"])
    def delete():
        # Delete a video game from the database by id
        game_id = request.form.get("game_id")
        Videogame.query.filter_by(game_id=game_id).delete()
        db.session.commit()
        return redirect("/")
```

Clicking on delete in front of any game will delete the game in question. To update the user has to change the genre of the game and hit update. the new genre will be added in the database replacing the old one.

e.g : Delete code

```
    @timed(enabled=True)
    @app.route("/delete", methods=["POST"])
    def delete():
        # Delete a video game from the database by id
        game_id = request.form.get("game_id")
        Videogame.query.filter_by(game_id=game_id).delete()
        db.session.commit()
        return redirect("/")
```

Decorators :

@Timed seen in the previous code snippets is a decorator that displays the name of the function and the time it's been running for. The logs given by this decorator are then registered in a logs txt file.

```
    def timed(enabled):
```

```
    def dec_timed(func):
        if enabled:
            def exec_time(*args, **kwargs):
                logger = create_logger()
                start = time.time()
                result = func(*args, **kwargs)
                end = time.time()
                logger.info("{} ran in {}s".format(func.__name__, round(end - start,
                                                                         2)))
                return result
            return exec_time
        else:
            return func
    return dec_timed
```

The decorator takes as argument *enabled* which gives the possibility to disable or enable the function.

When closing the application the database clears and closes in order not to have redundant data in the next use.

```
db.session.remove()
db.drop_all()
db.create_all()
```