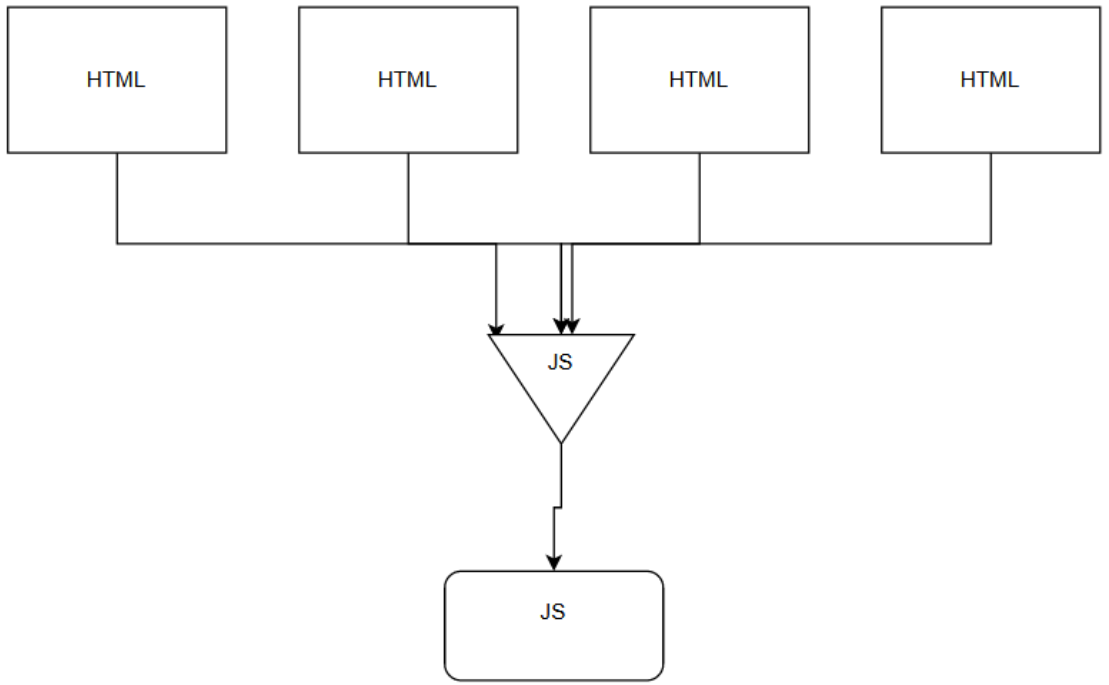

GALLERIA PIROLA
LASCIA CHE GLI OCCHI TI MOSTRINO LA POTENZA DELL'ARTE

SCHELETRO PRINCIPALE:



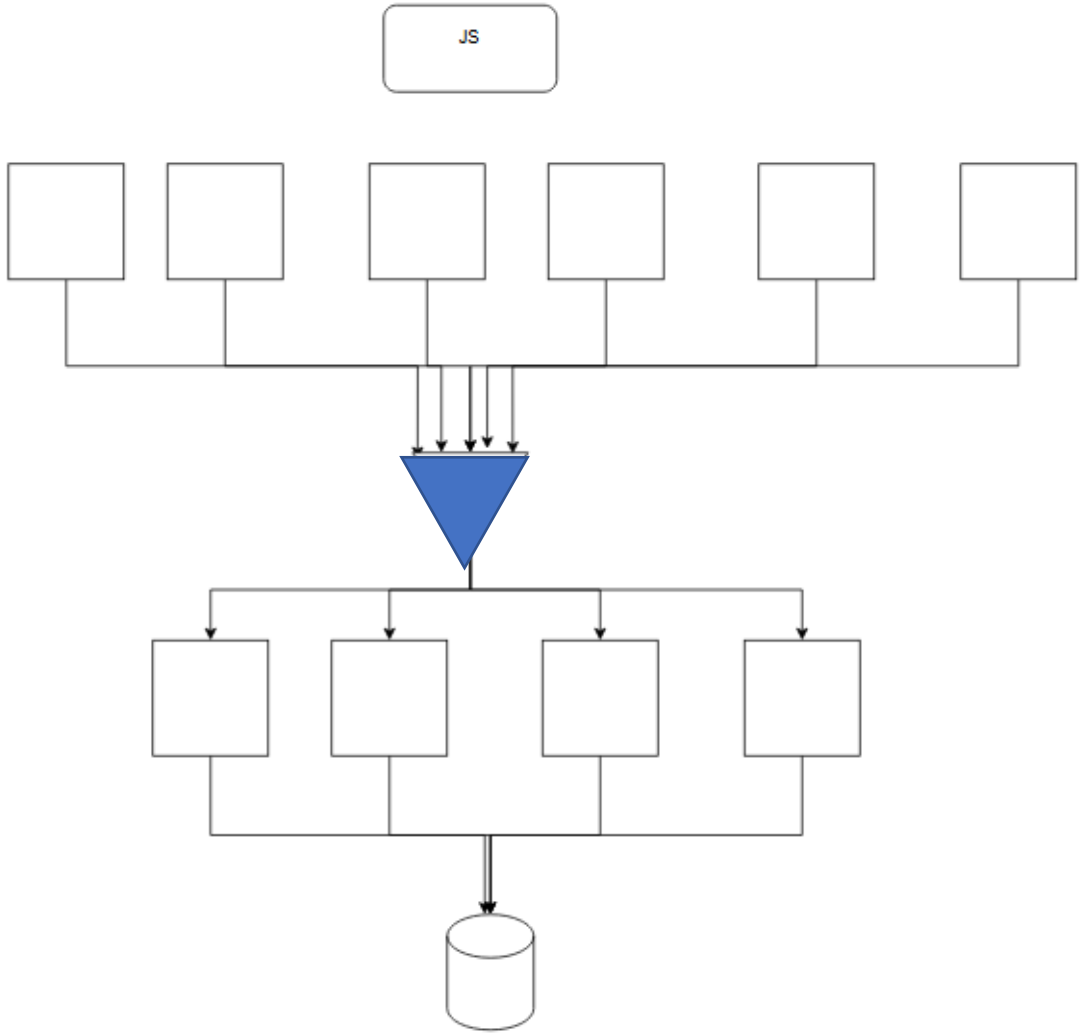
```
constructor(mainBox, navBar, sections, mair
    page('/', async () => {···
    });

    page('/login.html', async () => {···
    });

    page('/register.html', async () => {···
    });

    page('/opere.html', async () => {···
    });

    page('/user', async () => {···
    });

    page('/logout', async () => {···
    });

    page('/edit', async () => {···
    });

    page('/search',async()=>{

    })
    page();
}
```

```
import UserMan from './UserMan.js';
import PageMan from './PageMan.js';
import TicketMan from "./TicketMan.js";
import NewsMan from "./NewsMan.js";
import NewsLetterMan from './NewsLetterMan.js';
import MessageMan from './MessageMan.js';
import FeedbackMan from "./FeedbackMan.js";
import page from '//unpkg.com/page/page.mjs';
import CollectionMan from "./CollectionMan.js";
import FileSystemMan from "./FileSistem.js";
import OperaMan from "./OperaMan.js";
```

APP ENGINE:

- Utilizzo di Page Js
- Interazioni con FS
- Si appoggio a delle altre classi JS per gestire le interazioni con il server

SCHELETRO SECONDARIO:



CLASSI MANAGER

```
class CollectionMan {

    static getCollections = async () => {
        let response = await fetch('/getCollections', {
            method: 'POST',
        });
        const res = await response.json();
        console.log("AWAIT JSON", res);
        if (response.ok) {

            return res;

        }
    }
```

- Gestiscono le promise fornite dal server richiedendole in modalità asincrona.

CLASSI

```
ss Opera{
 constructor(title,description,artist,year,price,im
 {
     this.title=title;
     this.description=description;
     this.artist=artist;
     this.price=price;
     this.year=year;
     this.image=image;
 }
```

- Modellano gli oggetti necessari all’applicazione web

SERVER

```
app.post('/image', (req, res) => {
  daoOpera.getOperaPage(req.body.type).then((result) =>
    if (result.error) {
      res.status(404).json(result);
    } else {
      return res.json(result);
    }
  }).catch((err) => {
    res.status(500).json({
      'errors': [{ 'param': 'server', 'mgs': err }],
    });
  })
});
```

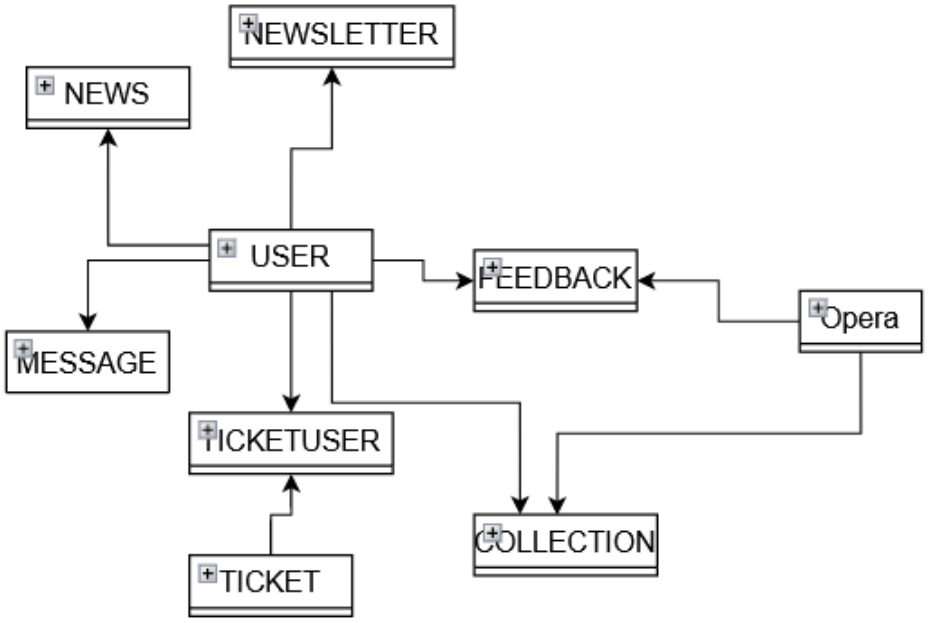
- Sono sviluppate le API REST
- Vengono eseguite le richiesta che provengono dalle classi manager
- Il server risponde fornendo delle promise .

DAO(DATA ACCESS OBJECT)

```
exports.getUser = function (username, password) {
  return new Promise((resolve, reject) => {
    const sql = 'SELECT name, surname, address,email,birthdayDate,username,passw
    db.get(sql, [username], (err, row) => {
      if (err)
        reject(err);
      else if (row === undefined)
        resolve({ error: 'User not found.' });
      else {
        const user = { name: row.name, surname: row.surname, address: row.addres
        let check = false;
        if (bcrypt.compareSync(password, row.password))
          check = true;
        resolve({ user, check });
      }
    });
  });
};
```

- Interrogano il database
- Restituiscono degli oggetti JSON

DATABASE


NEWSLETTER
NEWS
USER
FEEDBACK
Opera
MESSAGE
TICKETUSER
COLLECTION
TICKET