

06 - 06 - 2024

ONLY DATA

## Contenido

1.Introducció.....	2
2.Ferramentes i mètodes.....	2
3.Perspectiva estàtica.....	3
3.1.E/R (Entitat-relació).....	3
3.2. Col·leccions.....	4
} 3.3.CREATE.....	5
3.4.UPDATE.....	6
3.5. READ.....	7
3.6. Usuaris i Rols.....	7
4. Perspectiva dinàmica.....	8
4.2. Casos de Uso (Mètodes).....	9
5.1.Resultats obtinguts.....	11
5.2. Reflexions sobre el procés i possibles millores futures.....	11
6. Bibliografía y Webgrafía.....	12

# 1.Introducció

L'aplicació va relacionada amb una aplicació actual que en l'any 2023 es calcula va facturar 1000 milions d'euros. Aquesta aplicació s'anomena OnlyFans i és coneguda com "la xarxa social del sexe". En l'aplicació, models pugen contingut sexual, i els clients paguen per consumir-lo.

L'intenció d'aquesta aplicació és crear una base de dades on es recullen dades de models, els managers i les agències de managers. Per tal que models poden buscar manager, o manager models, sempre que no estiguin treballant amb ningú, per treballar junts. A la vegada que es guarden les agències per saber si els managers treballen en alguna agència o treballen per ells mateixa. També es guardaran els ingressos que té cada model i els ingressos que cada manager pot aconseguir que guanyen els models.

L'aplicació facilitaria la comunicació i la recerca de beneficis tan per part de models, com de managers i agències. On podrien contrastar els beneficis actuals amb els que podrien aconseguir. Disposa de les opcions de mostrar el managers que no tenen agència, per poder ser contractats per aquestes, i de mostrar models que no disposen de manager, per també poder ser contractades.

## 2.Ferramentes i mètodes

Les ferramentes usades per a crear l'aplicació son les següents:

- Draw.io: Ferramenta utilitzada en la que elaborarem el diagrama d'entitat relació de la nostra base de dades.

- Visual Studio Code: Aquesta ferramenta serà l'entorn de desenvolupament de l'aplicació. Serà on està programat el codi fet amb python.

- Tkinter: Aquesta llibreria s'ha importat al codi de python, que és la que permetrà crear l'interfície gràfica amb la qual s'interactua en l'aplicació.

- PyMongo: llibreria utilitzada per a poder utilitzar MongoDB en el codi de python.

-MongoDB: Serà el entorn de desenvolupament de la base de dades, la qual connectarem amb la nostra aplicació. S'ha usat aquesta aplicació per la facilitat de connexió entre el codi i la base de dades.

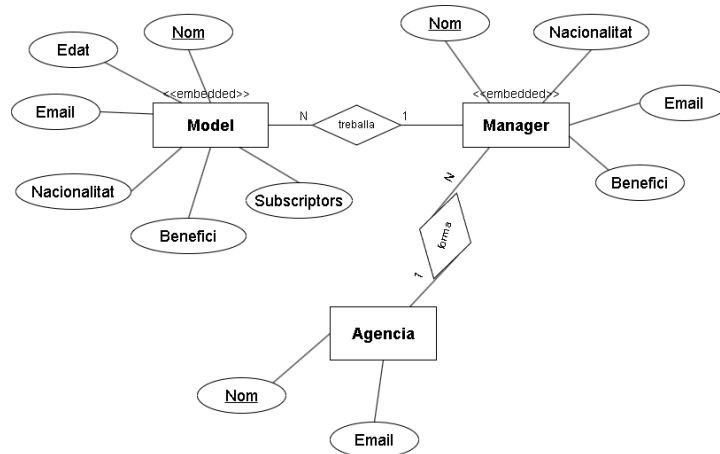
-Docker: S'ha utilitzat docker per a llançar una imatge d'un server de MongoDB, amb el qual connectarem el codi de python per a fer la base de dades.

### 3.Perspectiva estàtica

La perspectiva estàtica en el context d'un projecte de sistemes d'informació es refereix a la manera en què es descriuen les estructures estàtiques del sistema, incloent-hi els components principals i les seves relacions. Aquesta perspectiva és crucial per entendre com es compon el sistema i com les diferents parts interactuen entre elles sense considerar el comportament dinàmic o el flux de dades en temps real.

#### 3.1.E/R (Entitat-relació)

El model Entitat-Relació (E/R) és una representació gràfica que es fa servir per descriure les dades i les relacions entre les dades dins d'un sistema d'informació. Aquest model utilitza entitats, que són objectes o conceptes del món real amb significat dins del context del sistema, i relacions, que representen les associacions entre aquestes entitats.



**Fig. 1:** Diagrama d'entitat -relació

## 3.2. Col·leccions

El diagrama entitat-relació passa a col·leccions com es veu a continuació.

S'han creat 3 col·leccions, una per a cada entitat i amb els seus corresponents atributs.

### **#Col·lecció AGENCIA:**

```
{  
  
  "_id": "NomID",  
  
  "Email": "email@agencia.com"  
  
}
```

### **#Col·lecció MANAGER:**

```
{  
  
  "_id": "NomID",  
  
  "AgenciaID": "AgenciaNomID",  
  
  "Email": "email@manager.com",  
  
  "Nacionalitat": "nacionalitat",  
  
  "Benefici": 1000,  
  
  "Agencia": {  
  
    "$ref": "AGENCIA",  
  
    "$id": "AgenciaNomID"  
  
  }  
  
}
```

### **#Col·lecció MODEL:**

```
{  
  
  "_id": "NomID",  
  
  "ManagerID": "ManagerNomID",  
  
  "Edat": 25,  
  
  "Email": "email@model.com",  
  
  "Nacionalitat": "nacionalitat",  
  
  "Benefici": 500,  
  
  "Subscriptors": 1000,  
  
  "Manager": {  
  
    "$ref": "MANAGER",  
  
    "$id": "ManagerNomID"  
  
  }  
  
}
```

### **3.3.CREATE**

En aquesta part es mostra com es defineix l'estructura de la base de dades i es crea en el codi.

```
db = client['mydatabase']  
  
agencia_collection = db['agencia']  
  
manager_collection = db['manager']  
  
model_collection = db['model']
```

### 3.4.UPDATE

Ací es mostra la part del codi on manipulem dades, es veu les insercions que fem a les taules.

#### **# Inserir una agència**

```
agencia = {'nom': 'Agencia A', 'email': 'agenciaa@example.com'}
```

```
agencia_collection.insert_one(agencia)
```

#### **# Inserir un mànager**

```
manager = {
```

```
'agencia_id': ObjectId('60d5ecf1c2e54d4e3d2b7437'), # ID de la agencia
```

```
'nom': 'Manager 1',
```

```
'email': 'manager1@example.com',
```

```
'nacionalitat': 'Nacionalidad A',
```

```
'benefici': 'Beneficio A'
```

```
}
```

```
manager_collection.insert_one(manager)
```

#### **# Inserir un model**

```
modelo = {
```

```
'manager_id': ObjectId('60d5ecf1c2e54d4e3d2b7438'), # ID del manager
```

```
'edat': 25,
```

```
'nom': 'Modelo 1',
```

```
'email': 'modelo1@example.com',
```

```
'nacionalitat': 'Nacionalidad M',
```

```
'benefici': 'Beneficio M',

'subscriptors': 5000

}

model_collection.insert_one(modelo)
```

### 3.5. READ

Aquest punt es mostren les consultes a la base de dades, en les que mostrem les consultes de les models que no tenen manager per poder buscar per contractar. I també els managers sense agència. Com també mostrar totes les agències, models i mànagers.

```
data = agencia_collection.find()

data = model_collection.find()

data = manager_collection.find()

data = model_collection.find({'manager_id': None})

data = manager_collection.find({'agencia_id': None})
```

### 3.6. Usuaris i Rols

En aquesta part es mostra la part de codi on es fa el control d'usuaris amb els permisos pertinents. L'usuari sols podrà veure informació i l'administrador podrà veure-la i introduir-la.

#### **# Crea un usuari amb rol de lectura i escritura**

```
db.command("createUser", "username", pwd="password", roles=["readWrite"])
```

#### **# Crea un usuari sols amb rol de lectura**

```
db.command("createUser", "readonlyuser", pwd="password", roles=["read"])
```

#### **# Asignar el rol de lectura i escritura a un usuario existent**

```
db.command("grantRolesToUser", "username", roles=["readWrite"])
```



## # Revocar el rol de lectura y escritura a un usuario

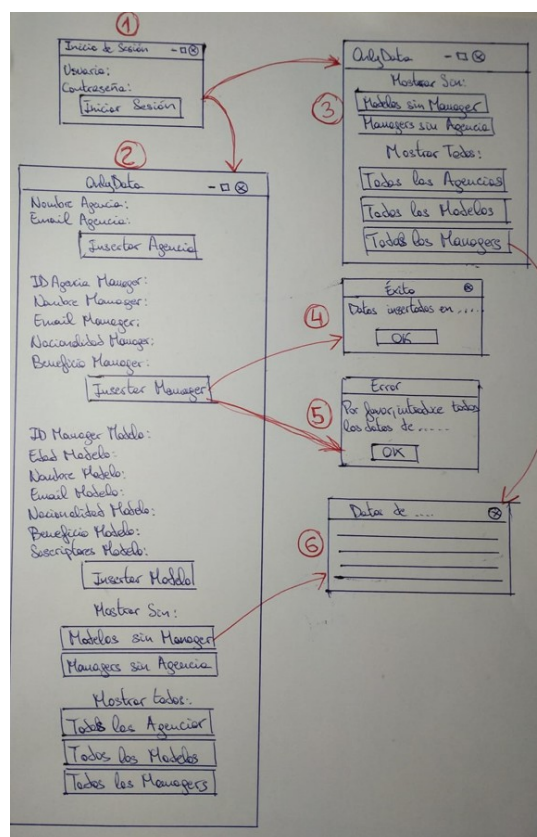
```
db.command("revokeRolesFromUser", "username", roles=["readWrite"])
```

## 4. Perspectiva dinàmica

La perspectiva dinàmica del projecte se centra en com els usuaris interactuaran amb l'aplicació a través de diverses finestres i funcionalitats. Aquesta part descriu l'aspecte i el comportament de l'aplicació, detallant les interfícies d'usuari i els casos d'us principals.

### 4.1.Sketch

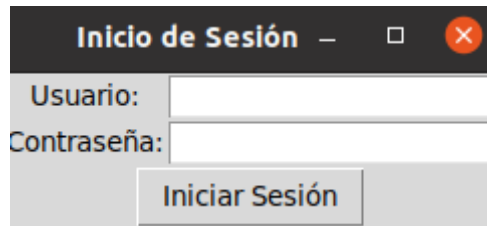
Esbos del que serà l'aspecte de l'aplicació on s'interactuarà amb ella. El disseny que es desitja obtenir, amb els camps per introduir de cada entitat i els botons respectius per inserir. Com també els botons per mostrar després agències, models i mànagers. I també els managers sense agència i els models sense manager.



**Fig. 2:** Sketch

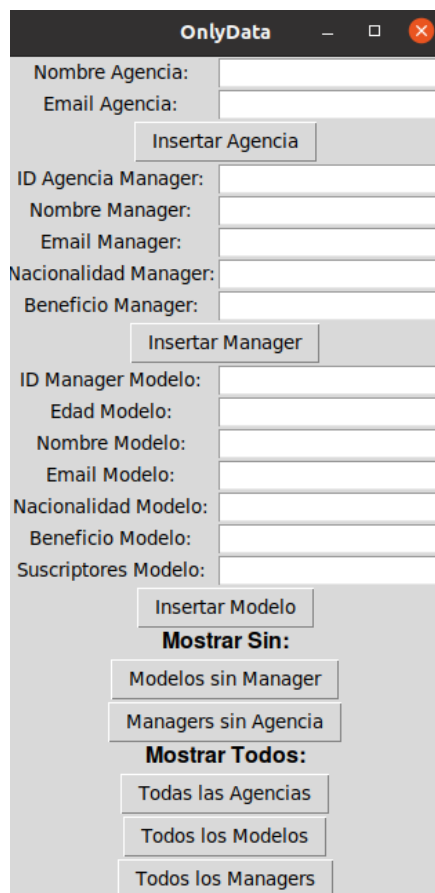
## 4.2. Casos de Uso (Mètodes)

**-Finestra 1:** En la finestra numero 1 hi ha 2 opcions. Iniciar sessió com a administrador o com a usuari. Si s'inicia com a administrador passaria a la finestra 2 i si s'inicia com a usuari passariem a la finestra 3.



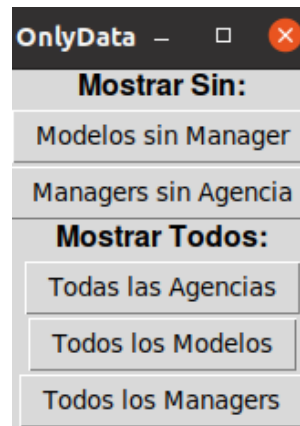
**Fig. 3:** *Finestra 1 del menú.*

**-Finestra 2:** Aquesta és la finestra que veuria l'administrador, on podrà introduir totes les dades i també mostrar-les.



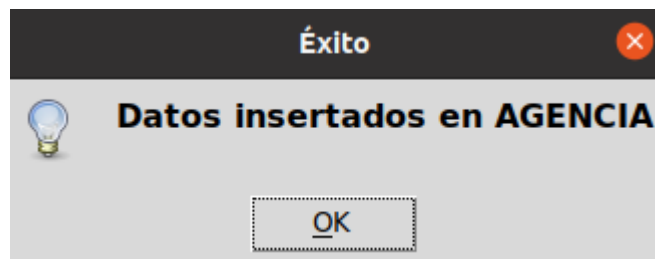
**Fig. 4:** *Finestra 2 del menú.*

**-Finestra 3:** Aquesta és la finestra que veuria un usuari, on no podrà introduir dades, tan sols mostrar-les.



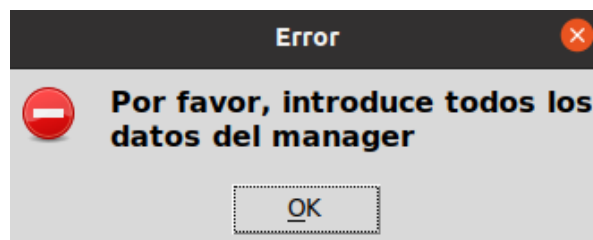
**Fig 5:** *Finestra 3 del menú.*

**-Finestra 4:** La finestra de “Éxito” apareixeria quan al introduir dades s’ha fet correctament introduint les dades necessàries.



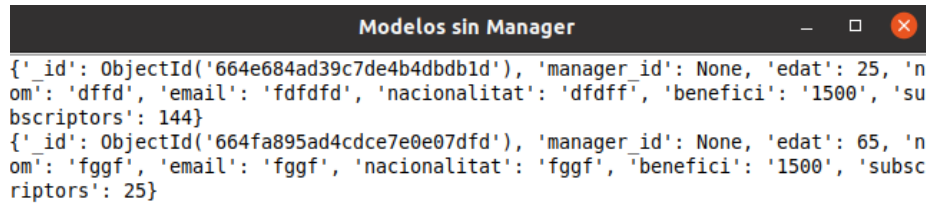
**Fig. 6:** *Finestra 4 del menú.*

**-Finestra 5:** La d’error apareix quan al introduir dades ens hem deixat algun camp necessari o no hem introduït alguna dada del tipus corresponent.



**Fig. 7:** *Finestra 5 del menú.*

**-Finestra 6:** És la finestra que apareix quan es clica un botó de mostrar dades.



**Fig. 8:** *Finestra 6 del menú.*

## 5. Conclusions

En aquesta secció, presentem una síntesi dels resultats obtinguts durant el desenvolupament del projecte, així com una reflexió sobre el procés seguit i possibles millores per al futur.

### 5.1. Resultats obtinguts

Referent als resultats obtinguts podem concluir que tenim la capacitat per poder realitzar un treball com aquest, amb un codi en python que cree una base de dades.

El treball l'hem realitzat de forma simple i senzilla. Amb 3 taules amb els atributs corresponents. El codi ha sigut simple per crear la base de dades i poder consultar -los. El resultat ha sigut sobretot d'aprenentatge amb el tema de documentació d'un projecte.

### 5.2. Reflexions sobre el procés i possibles millores futures

Durant els procés m'he donat conter que podria seguir aplicant funcionalitats al codi per tractar la base de dades. Com per exemple afegir mes botons de mostrar dades, per tindre-ho tot més clar i més visible per buscar qualsevol cosa. També afegir opcions per eliminar o modificar dades, etc...

També en quant al front-end s'hagués pogut millor molta cosa, fins afegir imatges o logotip per fer l'aplicació més atractiva.

Una bona planificació també és molt important, una possible millora en el futur seria una bona organització del pas a pas a realitzar.

## 6. Bibliografía y Webgrafía

-Enllaç Github:

<https://github.com/BernabeuCarles/MiniProyecto-II.git>

-Docker compose:

[https://github.com/docker/compose/releases/download/1.29.2/docker-compose-\\$\(uname -s\)-\\$\(uname -m\)](https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m))