



# OurWhats

Application web de messagerie instantanée  
réalisée avec Flask

Robin ARMINGAUD & Maxime GARNIER

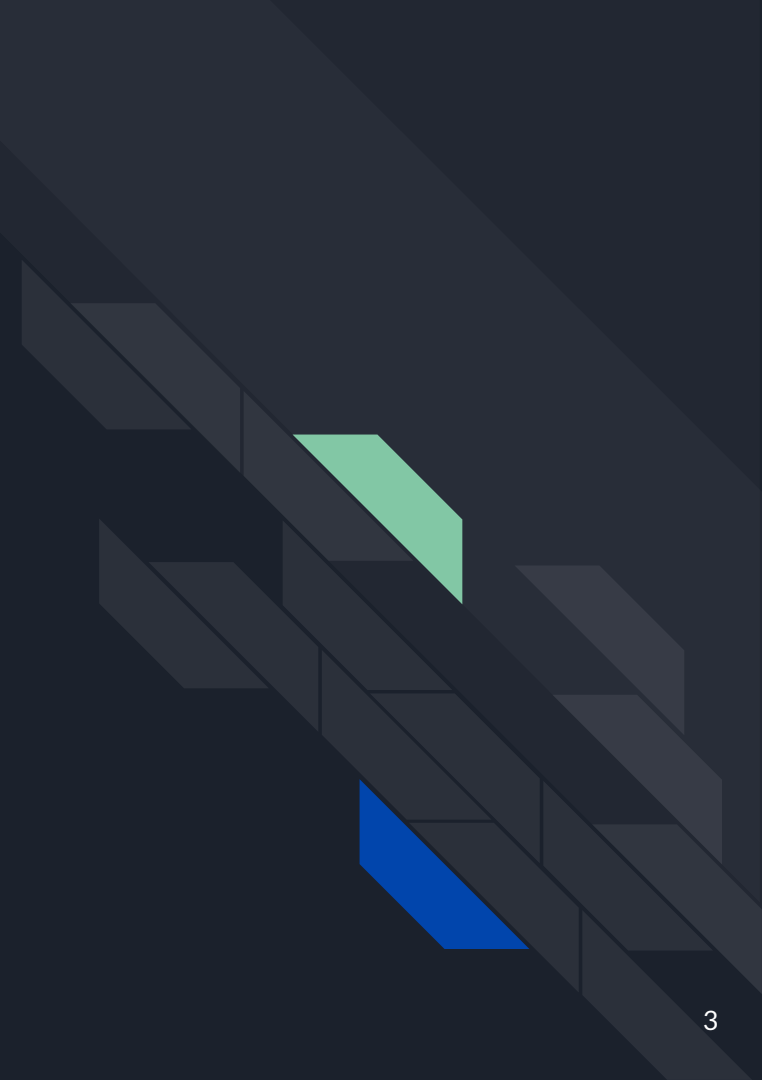


# Sommaire :

## Les fonctionnalités de notre OurWhats

1. Architecture de la base de données
2. La vue principale
3. Pièces jointes (et photos de profil)
4. Système d'authentification
5. Pistes d'amélioration technologique

# Architecture de la base de données



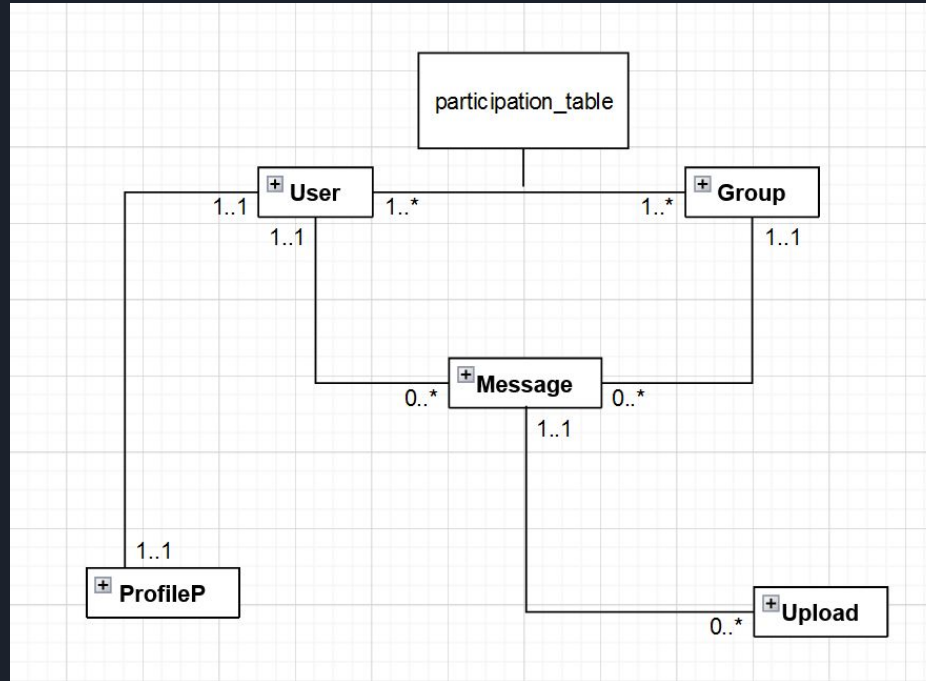



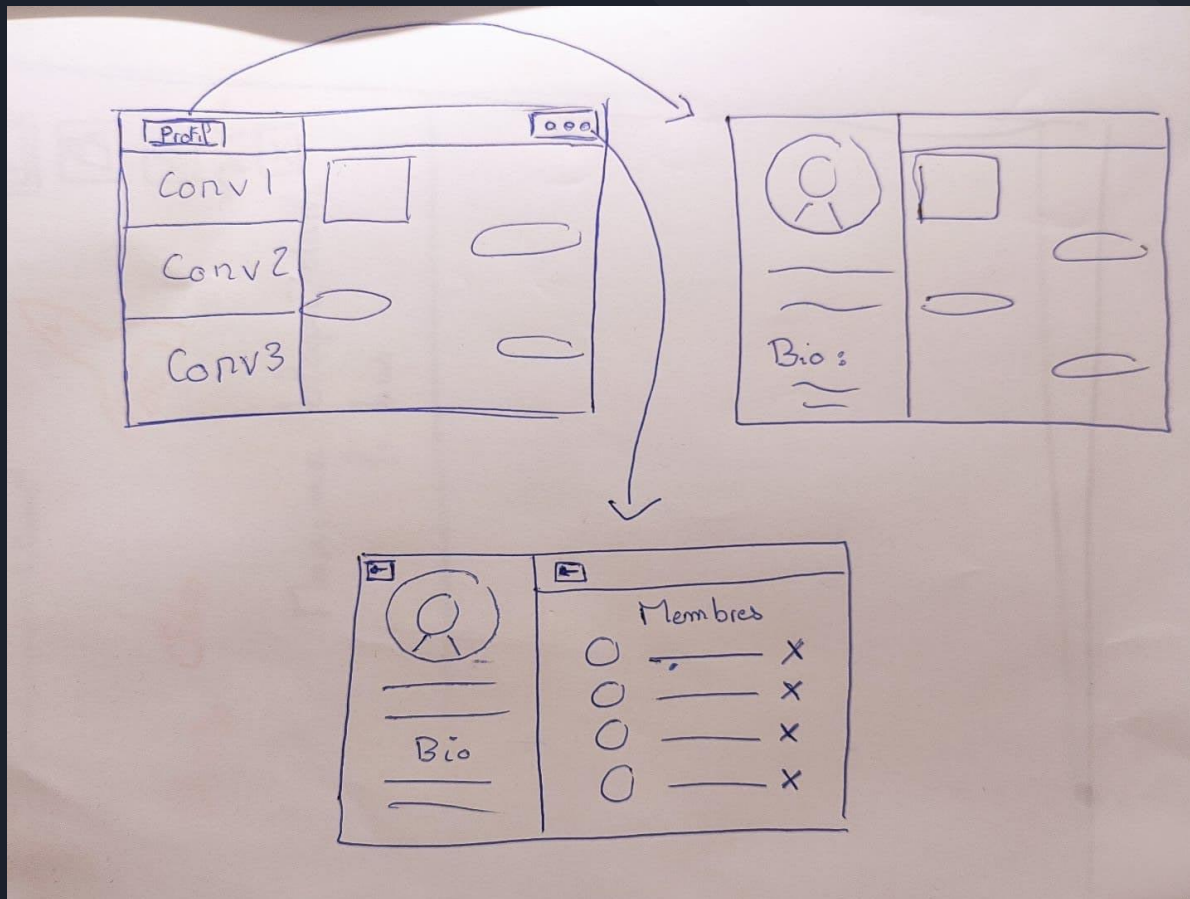
Diagramme de classes simplifié de la BDD



WHERE		ORDER BY	
	group_id	user_id	last_read_time
1	1	0	0001-01-01 00:00:00.000000
2	1	1	2022-03-21 21:58:18.650361
3	2	0	0001-01-01 00:00:00.000000
4	2	2	2022-03-21 22:08:29.270901
5	3	0	0001-01-01 00:00:00.000000
6	3	2	2022-03-21 22:08:29.817563
7	2	1	2022-03-21 22:11:01.129084
8	4	0	0001-01-01 00:00:00.000000
9	4	2	2022-03-21 22:08:30.461648
10	5	0	0001-01-01 00:00:00.000000
11	5	3	0001-01-01 00:00:00.000000
12	4	1	2022-03-21 22:11:01.537702
13	4	3	0001-01-01 00:00:00.000000

La table relationnelle *participation\_table*  
enregistre les dernières consultations

La vue principale



Croquis de travail

robin



TAF DCL



Rechercher un message

Quitter



Rechercher une conversation

1 message(s) non lus



TAF DCL

2022-03-21

robin : Tiens, voilà un fichier ZIP

3 message(s) non lus



Démarrage

2022-03-21

maxime : Salut

0 message(s) non lus



Démarrage

2022-03-21

OurWhats : Bienvenue dans OurWhats !



OurWhats

Nouveau groupe !

maxime

Bonjour à vous !



On a du travail pour demain ?

maxime

ça marche

Regarde, je suis dans l'espace



Non, mais on a le compte-rendu final à préparer pour la semaine prochaine

Wow

Tiens, voilà un fichier ZIP

[ue\\_web\\_example-authentication\\_example.zip](#)

Votre message

Parcourir...

Aucun fichier sélectionné.

Envoyer





## Grille CSS composée de 5 panneaux

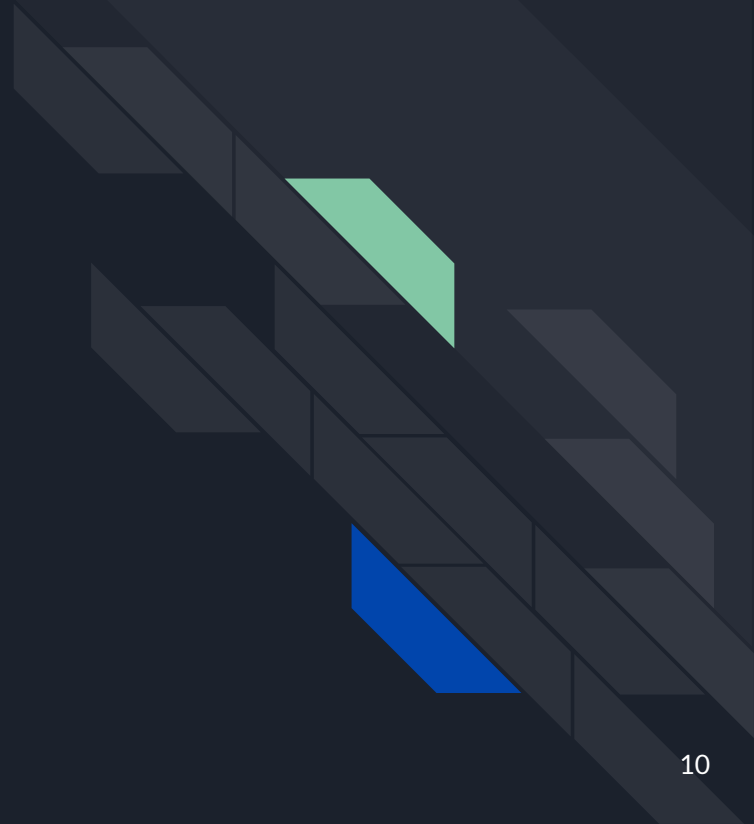
- .corner-menu
- .group-list
  - aperçu du dernier message du groupe
  - badge messages non lus
- .top-bar
- .msg-section
- .bottom-bar

Utilisation de plusieurs “modal boxes”, efficaces du point de vue UI.

Le côté “patchwork” est dû à nos différentes expérimentations.

La vue agrège les messages par “blocs” pour les restituer visuellement.

Pièces jointes  
(et photos de profil)



```
341 def send_attachments(request, message):
342     attachments = request.files.getlist('file[]')
343     for file in attachments:
344         if file and allowed_file(file.filename):
345             filename = secure_filename(file.filename)
346             file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
347             create_upload(message=message,
348                           filename=filename)
349     return
```

```
UPLOAD_FOLDER = 'static/uploads'
PP_FOLDER = 'static/profile_pics'
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png', 'gif', 'pdf', 'zip', 'mp4'}
ALLOWEDPP = {'jpg', 'jpeg', 'png', 'gif'}
app = flask.Flask(__name__)
```

```
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['PP_FOLDER'] = PP_FOLDER
```




Si la PJ n'est pas une image, l'affiche sous forme de lien.

Tout message doit comporter du texte.

La miniature est dimensionnée en fonction de la taille de la fenêtre.

La gestion backend des photos de profil est similaire.

# Systeme d'authentification




Utilisation du module *flask\_login* et suivi d'un tutoriel

<https://realpython.com/using-flask-login-for-user-management-with-flask/>

Pour simplifier l'utilisation durant le développement, nous avons fait le choix de ne pas implémenter de mot de passe, ce qui peut être fait aisément.

# Pistes d'amélioration technologiques

- 
- Mise à jour en temps réel : utilisation de web sockets
  - Orienter le client sur l'utilisation de requêtes AJAX, puis designer l'UI sur cette base
  - Restreindre l'utilisation de framework Bootstrap
  - Optimiser les requêtes sur la base de données en évitant de tout parcourir à chaque fois



Des questions ?