Rapport final Green Flag

Bettina-Sarah Janesch

Vincent Fournier

Présenté à Jean-Christophe Demers

4 2 0–C 6 1– I N

19 décembre 2024

Présentation générale

Greenflag est une application web de rencontres qui emploie des algorithmes afin de trouver des partenaires avec des intérêts similaires. Elle permet également la communication entre les utilisateurs en temps réel et la possibilité de signaler un usager pour mauvais comportement. Il y a également un système de notification connectée avec le mécanisme de matching et chat, ainsi qu'une gestion de session qui génère des jetons JWT qui rend l'application plus sécuritaire.

Les fonctionnalités attendues sont tous là.

Résumé du développement pendant la session

1. Respect ou modification des objectifs – Difficultés rencontrées.

Vincent Fournier

- La compréhension et le codage de l'algorithme Meanshift a été plutôt difficile. Cela m'a pris plus de temps que planifier dans notre planification.
- L'implémentation de librairie dans le frontend React n'a pas été simple et au lieu de nous sauver du temps cela à plutôt fait le contraire. Certaines librairies ne fonctionnaient pas avec tailwind ou fonctionnaient après quelques manipulations. L'usage de la librairie react-hook-form ne fonctionnait pas avec n'importe quelle librairie de composantes externes comme MUI.

Bettina-Sarah Janesch

Les difficultés surmontées ont été dans la base de données, backend et frontend.

- DAO & AccountManager: Les difficultés rencontrés incluent, entre autres, transférer des images du frontend au backend et après dans notre base de données clé-valeur et ensuite seulement la clé dans postgres, et aussi le fait que DAO, AccountDAO & AccountManager ont été la 1ere itération de notre communication avec la base de données. Coder des fonctions et vues dans postgres pour me faciliter la tâche n'étaient pas évidents non plus.
- La page des suggestions (MatchingPage) a été refactorisé au moins 5 fois. Au début, j'avais faire la composante de carte profil qui affiche correctement ce que le backend envoie. Après, pouvoir modifier la page et la requête vers le backend pour afficher tous les profils suggérés. De suite, attacher la librairie 'tinder react card' était particulièrement difficile et j'ai pris quelques jours pour la faire marcher et déboguer. Ensuite, j'ai retravaillé la logique de la page : quand on arrivait à la fin de la liste des profils disponibles dans le frontend, rien ne se passait. Donc j'ai refactor encore pour pouvoir redemander des nouveaux profiles et que ça soit quand même fluide visuellement aussi.

- Système d'authentification : difficulté de décortiquer la logique des jetons JWT et comment les régénérer, vérifier leur validité etc.

Fonctionnalités

Fonctionnalités parfaitement fonctionnelles :

- Création de compte de base
- Possibilité de se connecter à l'application
- Pouvoir se déconnecter de l'application
- Gestion sécurisée d'une session avec des JWT (jetons web JSON)
- Modifier caractéristiques du profil
- Ajout des photos sur le profil
- Accéder seulement aux pages protégées quand on est connecté
- Vice-versa : aucun accès sur les pages publiques par un usager connecté
- Impossibilité d'accéder sur la page de suggestions avant que le profil soit complété - pour que l'algorithme ait toutes les informations pour bien trier les suggestions
- Page de suggestion : pouvoir swiper des profils, a l'épuisement local de profils l'application continue d'en retrouver dans la base de données pour une expérience quasiment fluide
- Algorithme Meanshift: Algorithme non-supervisé de regroupement qui permet de trouver des usagers à suggérer selon leurs champs d'intérêts et leurs préférences
- Page de chatrooms: Pouvoir voire une liste des matches/discussions avec d'autres usagés (incluant le dernier message) et dit être redirigé
- Page individuelle de chat : Possibilité de discuter en temps réel avec d'autres utilisateurs
- Sélecteur de thèmes visuels : choisir entre plusieurs thèmes de couleurs qui change l'entièreté de l'apparence de notre application
- Paramètres compte : visualiser et modifier son profil
- Paramètres compte : choisir algorithme de tri qui influence les suggestions, modifier mot de passe, effacer profil
- UserFactory et Simulation de swipe : créer un nombre désiré de faux utilisateurs et de les faire accepter ou refuser des suggestions, ainsi que de remplir les fakes conversations de messages

Fonctionnalités partiellement fonctionnelles :

- Système de notifications
- Système de redflag

Identification des éléments techniques

Interfaces graphiques

Notre conception initiale a peu changé, elle suit nos maquettes. Il y a plutôt des différences subtiles, et aussi la page des paramètres du compte, les notifications et thèmes.

Données persistantes

Base de données utilisée : Postgres. Elle est utilisée amplement dans notre backend également pour écriture et lecture.

Exemple lecture : dev\backend\DAOs\ChatDAO ligne 81

Exemple écriture : dev\backend\DAOs\AccountDAO ligne 111

Structures de données

Dictionnaire : \dev\backend\Simulation\user_factory.py ligne 144

Tuple: \dev\backend\Managers\account_manager.py ligne 49

Numpy ndarray: \dev\backend\Algorithms\algo_meanshift.py lignes 35, 55, 57

DischargedList qui compte aussi comme bibliothèque, détaillé plus bas.

Patrons de conception

Nous avons codé les 4 patrons dans notre projet.

- AlgoStrategy (Vincent Fournier) : Dans ce cas, Strategy est utilisé comme pont pour faciliter l'usage de différent Algorithme. Se trouve dans \dev\backend\Algorithms\algo_strategy.py

- Observer (Vincent Fournier) : Observer permet d'envoyer des messages vers la base de données lorsqu'on en a trop ou que l'on en a depuis trop longtemps dans notre DischargedList. Se trouve dans \dev\backend\util_classes\discharged_list.py à la ligne 21

- Factory + SwipingStrategy (Bettina-Sarah Janesch) : Nous avons fait une usine de faux usagers pour ajouter des données à notre application qui enrichit l'expérience usager. Tous ces faux usagers emploient des stratégies de swiping pour faire (ou non) des matchs.
  \dev\backend\Simulation\ user_factory.py; test_simulator.py; swiping_strategy.py
- DAO (Bettina-Sarah Janesch) : notre classe qui communique avec la base de données.
  \dev\backend\DAOs\dao.py

Bibliothèque

Bettina-Sarah Janesch

- fetchData: dev\frontend\src\api\fetchData.ts

Fonction utilitaire générique pour effectuer des requêtes POST avec Axios, traitant les réponses JSON ou images, via une interface T, sous forme de promesse type T.

Un exemple de son usage :

```
const useTriggerFetch = <T extends any[]>(
  { url, data: incomingData }: IUseFetch,
  trigger: number
) => {
  const [data, setData] = useState<T | null>(null);
  const [error, setError] = useState("");
  const [loading, setLoading] = useState<boolean>(true);

  useEffect(() => {
    setLoading(true);
    setData(null);
    fetchData<T>(url, incomingData)
      .then((data) => {
        setData(data);
        console.log("data", data);
      })
      .catch((error) => {
        console.error(error);
        setError(error);
      })
      .finally(() => {
        setLoading(false);
      });
  }, [trigger]);

  return {
    data,
    loading,
    error,
  };
};

export default useTriggerFetch;
```

- useTriggerFetch: dev\frontend\src\api\useTriggerFetch.ts

Hook personnalisé et générique permettant de déclencher des requêtes HTTP avec fetchData à l'aide d'un declencheur. Un exemple de son usage :

```
const MatchingPage: React.FC = () => {
  const [refetchTrigger, setRefetchTrigger] = useState(0);
  const algo = sessionStorage.getItem("algo") || "Meanshift";
  const [cardsSwiped, setCardsSwiped] = useState<
    { suggestion_id: string; swiped: boolean }[]
  >([]);
  const {
    data: profileData,
    loading: profileLoading,
    error: profileError,
  } = useTriggerFetch<IProfileData[]>(
    {
      url: "/suggestions",
      data: { id: sessionStorage.getItem("id"), algo: algo },
    },
    refetchTrigger
  );

  const handleEndOfList = () => {
    setRefetchTrigger((prev) => prev + 1);
  };
```

- IconButton: dev\frontend\src\components\IconButton.tsx

Un bouton générique avec l'image d'une icône, qui peut accepter une page (string) pour naviguer vers une page, ou sinon, un callable qui exécute une fonction.

Un exemple de son usage :

```
return (
  <div className={`w-full h-14 pb-6 ${classname || ""}`}>
    <div className="flex w-full justify-evenly  bg-primary-color">
      <IconButton icon={matchingIcon} page="matching" />
      <IconButton icon={messageIcon} page="chatrooms" />
```

```
      <IconButton
        className="h-28 w-28"
        icon={RedFlag}
        onClick={() =>
          SwipeLeft(profile.suggestion_id, index === 0)
        }
        suggestion_id={profile.suggestion_id}
      />
```

*Vincent Fournier * Il s'agit également de notre structure de donnée à créer*

DischargedList est une liste auquel on attribue des observateurs qui recevront son contenu lorsque celle-ci aura été remplie d'au moins un objet pendant un certain temps donné ou atteint une limite donnée. Le code se trouve à \dev\backend\util_classes\discharged_list.py

Un exemple de son usage : À chaque message envoyé dans le chat il est gardé dans la DischargedList self.requests et lorsque celle-ci a atteint 5 messages ou qu'elle possède un message depuis 5 secondes elle est déchargée dans le __call__ de l'Oberser ChatroomManager qui envoie son contenu à la base de données.

```
dev > backend > Managers > chatroom_manager.py > ChatroomManager > get_chatrooms
18  class ChatroomManager(DischargedList.Observer):
19      def __init__(self) -> None:
20          self.requests = DischargedList(5,5)
21          self.requests.add_observer(self)
22
23      def __call__(self, list:list[tuple[str,int]]):
24          response = ChatDAO.send_messages(list)
25          if response:
26              return response
27          return False
```

```
dev > backend > Managers > chatroom_manager.py > ChatroomManager > get_chatrooms
18  class ChatroomManager(DischargedList.Observer):
81      def add_chatroom_message(self, data) -> None:
82          self.requests.add_item(data)
```

Algorithme

MeanShift est un algorithme de clustering non supervisé auquel il nous s'agit juste de données une matrice de points de n dimensions. Le code se trouve à \dev\backend\Algorithms\algo_meanshift.py

Mathématiques

Nous avons implémenté la formule haversienne pour calculer la distance entre 2 points géographiques, directement dans la base de données. Le code se trouve à \dev\database\views_functions.sql à la ligne 136 (fonction calculate_distance)

Exemple : calculer la distance d'une personne en comparant avec la position actuelle de l'usager de l'application:

dev > database > views_functions.sql

```
157  DROP FUNCTION fetch_distance;
158  CREATE OR REPLACE FUNCTION fetch_distance(logged_id INT, suggestion_id INT)
159  RETURNS NUMERIC
160  AS $$
161  DECLARE
162      lat1 DOUBLE PRECISION;
163      long1 DOUBLE PRECISION;
164      lat2 DOUBLE PRECISION;
165      long2 DOUBLE PRECISION;
166      distance DOUBLE PRECISION;
167      second_user_id INT;
168  BEGIN
169      SELECT last_lat, last_long
170      INTO lat1, long1
171      FROM member
172      WHERE id = logged_id;
173
174      SELECT member_id_2
175      INTO second_user_id
176      FROM suggestion
177      WHERE id = suggestion_id AND member_id_1 = logged_id;
178      IF second_user_id IS NULL THEN
179          RETURN NULL;
180      END IF;
181      SELECT last_lat, last_long
182      INTO lat2, long2
183      FROM member
184      WHERE id = second_user_id;
185      IF lat1 IS NULL OR long1 IS NULL OR lat2 IS NULL OR long2 IS NULL THEN
186          RETURN NULL;
187      END IF;
188      distance := calculate_distance(lat1, long1, lat2, long2);
189      RETURN ROUND((distance / 1000)::NUMERIC, 1);
190  END;
191  $$ LANGUAGE plpgsql;
192
```

Regex

Notre regex sert à s'assurer que le courriel donné par l'usager est conforme à une structure d'adresse courriel. Le code se trouve dans \dev\frontend\components\form_components\RegisterForm.tsx à la ligne 106

Améliorations possibles

Système de notification

En ce moment, une notification est générée quand un match est créé ou un message reçu. Le problème arrive quand on décide de ne pas cliquer sur la notification pour y accéder, et a la place d'aller nous même dans la liste de conversations et voir le message. Le système n'est pas conçu pour savoir ça, et la notification reste là. Un cas limite observé et non traité par manque de temps.

Frontend de l'application

Notre application contient clairement ses forces dans le backend et dans la base de données complexe et performante. Malgré beaucoup de temps passé dans le frontend pour faire marcher les données reçues, le visuel a été mis en dernière priorité et peaufiné à la fin seulement. Il y a aussi des places dans le code où l'organisation aurait pu être faite autrement pour améliorer l'architecture et la réutilisation des composantes. Par exemple, l'icône et aussi le bouton contenant une icône sont des composantes génériques et réutilises partout, pendant que le bouton principal a été copié partout au lieu de faire une composante générique. Le form du questionnaire est également un élément du frontend qui pourrait être améliorer sa structure est bien trop rempli pour une seule composante. Il serait intéressant de le morceler en plusieurs composante réutilisables et plus génériques.

Email adapter

On aurait voulu coder le patron de conception adapter et introduire une bibliothèque externe pour envoyer des courriels de confirmation d'une adresse électronique quand quelqu'un crée un nouveau compte. Cela empêcherait en réalité la création des faux comptes et contribuera à la crédibilité de notre application.

DAO

Quels sont les éléments techniques que vous pourriez améliorer. On ne parle pas ici des ajouts de fonctionnalité possibles (il y en reste toujours beaucoup), mais plutôt des améliorations techniques de votre projet. Par exemple, telles portions de code devraient être réécrites avec telles approches, utiliser telles librairies plutôt qu'une autre, modulariser telles portions du code, améliorer telles parties du design puisque peu extensibles, utiliser tels algorithmes ou structures de données, etc.

Système de Redflag

Notre système de Redflag est très rudimentaire. Pour l'instant, il n'affecte notre algorithme de suggestion que par la quantité de flag qu'un utilisateur a obtenu. Il serait intéressant de donner des poids aux raisons du pourquoi l'usager a été flagger et que la somme de ces poids influence l'algorithme.

Find_suggestion dans MatchingManager

La méthode find_suggestion est trop longue et n'est pas assez générale. Elle tente de palier des cas limites qui n'avaient pas été envisagé, mais le fait de façon trop direct et pas assez malléable. Elle fait du traitement des données reçues qui devrait probablement être fait à l'extérieur de celle-ci.

Auto-évaluation individuelle

Évaluation Bettina-Sarah Janesch

1. Notre formation collégiale a été marqué par l'apprentissage vertigineux de plusieurs langues de programmation et paradigmes de programmer, des frameworks, des patrons de conception, comment architecturer son code, l'importance de la modularité et généricité, des bonnes habitudes de programmeur comme tester souvent et rester organisé, comment gérer et planifier un projet logiciel d'ampleur considérable, le travail assidu en équipe, penser dehors d'une boite, et encore plus. Le projet synthèse a amplement profité de tous les outils acquis pendant le DEC et nous a amené vers une indépendance professionnelle que je ne m'y attendais pas. Nous avons utilisé plusieurs langues dans le projet, on a appris des frameworks anciennement inconnues, nous avons réussi de produire du code générique et on constate cette utilité a plusieurs endroits dans le projet. Les patrons de conception nous ont forcé de les comprendre profondément pour pouvoir les coder après. De plus, nous avons eu une approche systématique et on est resté organisé : tester souvent notre code a fait en sorte que nous avons une application robuste avec la plupart de fonctionnalités souhaitées au début et on a réglé les bugs progressivement au lieu d'arriver à la fin en désastre.
2. Je mets personnellement énormément de pression sur moi-même et j'ai des attentes souvent impossibles : l'aspect visuel de notre application m'a déçu un peu malgré l'effort qui a été investi. Nous avons accompli des objectifs incroyables et je suis tellement fière de tout. La somme de tous les compétences et connaissances acquises dans le DEC dépasse largement mes attentes au début du DEC. Faut mentionner qu'on a dépassé les attentes personnelles du projet grâce aux exigences demandés dans le cadre du projet et aussi pendant la session avec les discussions entretenues avec le professeur. Nous avons eu la chance d'être inspiré par des gens incroyables et beaucoup de soutien qui a favorisé l'apprentissage accéléré et l'accomplissent de la plupart de nos objectifs.
3. Il y a beaucoup d'éléments techniques qui m'ont surpris, mais apprendre et vraiment comprendre React et TypeScript sont 2 choses différentes. Je reconnais d'avoir seulement gratté la surface et j'ai hâte d'en découvrir plus. J'ai réalisé aussi qu'on sous-estime le temps nécessaire à dédier pour faire un site

web élégant. Sinon, je reconnais l'importance de bien modéliser un projet avant commencer coder. Notre vision interne rarement va se synchroniser avec le résultat. Il y aura toujours des imprévus qui faut gérer sur le coup, et avoir une bonne planification du projet de l'avance aide beaucoup.

4. Concernant l'effort et le temps investi, je me donne 100% car j'ai donné plus de temps qu'attendu, et pendant les soirées, les fins de semaines, et pendant mon travail, j'ai priorisé ce projet au détriment de tous les autres projets de la session. Pour la qualité des fonctionnalités livrés, je me donne 85-90%. J'ai accompli la plupart de mes objectifs et tout fonctionne comme attendu la plupart du temps. Pour la qualité du code, je me donne 85-90% aussi, j'ai quelques fichiers ou je suis très fière, notamment mes fichiers frontend useFetch, useFetchTrigger, MatchingPage, CustomTinderCard, et backend AccountManager, AccountManager & DAO, photoDAO et CryptKeeper (authentification). Je note ces fichiers car il y a eu de la difficulté variée : sois que c'etait le premier DAO codé, ou le fichier contient des composantes ou fonctions génériques, ou le code a été difficile à comprendre pour gérer le stockage d'images, ou la logique de l'authentification.

Évaluation Vincent Fournier

1. Notre projet représente bien le cumulatif de toutes les matières et sujets que nous avons vu lors de notre formation. En partant de la base, les langages de programmation. Nous avons 3 des 5 langages que nous avons appris lors du programme soit Pgsql/plpgsql, Python et Javascript (Typescript). Nous faisons l'usage de plusieurs modèles de conception, notamment l'Observer et la Stratégie. Nous avons même notre propre algorithme personnalisé. Il est important de remarquer aussi comment les cours de conceptions, nous ont aidé à procéder de façon plus structurée lors de notre développement. L'usage de SCRUMs, nous a permis de mieux réévaluer notre cheminement de développement et à mieux comprendre ce que notre collègue a fait de son côté. Dans l'ensemble, c'est impressionnant de voir à quel point nous avons appris d'information lors de ce DEC.
2. Je suis très satisfait de notre projet. Lors de la conception, j'éprouvais des doutes à savoir si on allait être capable de finir toutes les fonctionnalités que l'on voulait faire. Finalement, nous avons pu faire la totalité de celles-ci même si elles ne sont pas tous parfaites elles fonctionnent. Selon moi, c'est un exploit d'avoir fait autant en sachant qu'il y a 1 an et demi nous savions très peu de ce en quoi consiste la programmation et qu'il s'agit de notre premier projet entièrement pensé et conceptualisé par nous.
3. Je retiens que la conception d'un projet est la partie la plus importante de son développement. Une fois que celui-ci est bien entamé il n'est pas toujours facile de le modifier afin à ce qu'il réponde à un nouveau problème ou une nouvelle

fonctionnalité. S'il est bien conceptualisé, il est possible d'éviter de futures erreurs dans la structure du programme. J'ai appris aussi que le travail en équipe était un travail qui demande beaucoup d'effort et de patience et qu'il m'arrive de manquer de la dernière. J'ai la fâcheuse tendance à remarquer mes erreurs de communications qu'après les faits ce qui m'empêche d'éviter des conflits facilement évitables.

4. Entre 90% et 100%, j'ai eu un essoufflement pendant une semaine dans laquelle j'ai ralenti la cadence de mon codage. J'aurais également voulu produire du code plus lisible. J'ai l'impression que mon code est trop long et pas assez comportementalisé ou que j'ai de trop long nom de valeur. Je ne sais pas exactement ce qui cloche, mais je ne trouve pas que cela fasse propre. Sinon, je assez satisfait de mon Meanshift. Il m'a permis d'affiner mes connaissances en algorithmes. J'ai également adoré faire la DischargedList. Elle n'est pas bien compliquée, mais de la voir fonctionner dans notre application m'a donné une certaine satisfaction.

Auto-évaluation d'équipe

Avis Bettina-Sarah Janesch

Je pense que les deux on a contribué beaucoup en termes de fonctionnalités livrés et aussi l'effort investi dans le projet. Il y a eu une bonne collaboration en équipe et quand un n'était pas capable d'investir 100%, l'autre donnait 120%, et vice versa. Les deux on a eu des parties difficiles du projet à gérer. On a même réussi d'avancer le projet avec du pair-programming. Je suis contente qu'on a réussi organiser le projet pour que les deux on peut contribuer également dans le frontend, backend et base de données. Nous avons également eu des séances de 'brain storming' ensemble qui nous a aidé à décortiquer la logique des parties difficiles du code. Malgré la collaboration quand même excellente, il y a eu des moments ou la communication a pu être meilleure. A cause du temps manquant, des attentes du cours et du projet, nous avons ressenti beaucoup de pression et stress, et nous n'avons pas toujours bien géré nos émotions un envers l'autre. De mon côté j'aurais pu baisser les attentes parce que je sais que j'ai affecté mon coéquipier avec la pression. En revanche, nous avons eu des moments ou la communication efficace nous a aidé à surmonter des difficultés et mieux organiser notre travail. Le produit résultant dépasse les attentes à nous deux et nous sommes très fiers de notre livrable.

Avis Vincent Fournier

En voyant tout le travail que nous avons accomplis et le projet qui en abouti en résultante, je ne peux croire que nous ayons pu le faire d'une meilleure façon. Moi et ma coéquipière avons mis beaucoup d'efforts et de temps dans ce projet et selon moi à tempo inhumain. Notre séparation du travail était parfaite. Je crois que de séparer les tâches par fonctionnalité comme nous l'avons fait, à l'échelle de notre projet, est la

meilleure façon afin d’optimiser le temps développement. Bien que l’on ait chacun(e) à apprendre les plusieurs facettes de notre application soit Frontend, Backend et Base de données, il est plus pratique de faire le cheminement logique de bout en bout et de déboguer soi-même ce que chaque partie de logique envoi à la prochaine. Notre seul souci a été dans notre communication en équipe. Nous avons à plusieurs occasions manquées été éprit de trouble de communication. Il s’agit d’un problème seulement solvable individuellement.