Arnaud SIMON, Nathanaël NEZONDET-RENAUD, Hamza ESSAMAMI

Rapport projet annuel 4A

**Sommaire :**

Table des matières

[I - Présentation générale de notre projet 2](#_Toc45834141)

[II - Spécifications générales et techniques 4](#_Toc45834142)

[**Périmètre fonctionnel** 4](#_Toc45834143)

[**Arborescence du site** 5](#_Toc45834144)

[**Arbo Utilisateur classique** 5](#_Toc45834145)

[**Arbo Utilisateur administrateur** 5](#_Toc45834146)

[**Fonctionnalités pour chaque page** 6](#_Toc45834147)

[**Frontend** 6](#_Toc45834148)

[**Page play.html** 8](#_Toc45834149)

[**Page home\_user\_app.html** 9](#_Toc45834150)

[**Page profile\_user\_app.html** 9](#_Toc45834151)

[**Page game\_user\_app.html** 11](#_Toc45834152)

[**Back Office** 12](#_Toc45834153)

[III - Architecture technique 18](#_Toc45834154)

[**Schéma architecture technique** 18](#_Toc45834155)

[**Technologies choisies** 20](#_Toc45834156)

[IV - Planning et répartition des tâches 20](#_Toc45834157)

[V - Travail réalisé (tableau des tâches) et difficultés rencontrées 21](#_Toc45834158)

[**Tableaux des tâches** 21](#_Toc45834159)

[**Difficultés rencontrées** 24](#_Toc45834160)

[**Récupérer un dessin avec canvas** 24](#_Toc45834161)

[**Trouver des comment exploiter les données de quickdraw (traitement des données, création de modèles, …)** 24](#_Toc45834162)

[**Transformer les dessins utilisateurs pour qu’ils puissent être utilisés par nos modèles (et donc trouver les bonnes catégories)** 25](#_Toc45834163)

[**Créer des datasets au format .npy à partir des images des utilisateurs** 26](#_Toc45834164)

[**Guide pour parcourir notre projet (architecture du repo git, liens entre les fonctions, …)** 28](#_Toc45834165)

[**Architecture du repo git** 28](#_Toc45834166)

[**Arbre des commits** 31](#_Toc45834167)

[VI - Conclusion 35](#_Toc45834168)

[**Pistes d’amélioration** 35](#_Toc45834169)

[**Brève conclusion** 35](#_Toc45834170)

# I - Présentation générale de notre projet

**Nom du projet :** Learn2draw

**Titre :** Jeu d’apprentissage pour le dessin

**Courte description** :

Notre jeu a pour objectif d’apprendre aux jeunes enfants de 7 à 70 ans à améliorer leur technique de dessin tout en s’amusant. Notre jeu s’adressera aux priorités aux enfants et contiendra donc des récompenses amusantes pour stimuler les joueurs et leurs donner envie de s’améliorer.

**Fonctionnalités générales**:

* Dessiner
  + Temps différents (ex : 10 / 20 / 30 sec)
  + Indications données pendant le dessin (régulier selon nb traits, temps ou autre),

Propositions d’interprétation du dessin

* + Effacer tout le dessin pour recommencer / ou seulement une partie avec une gomme
  + Bouton valider pour « arrêter » le dessin
* Récompenses / Badges
  + Système de points / classement (le plus de points, la meilleur moyenne, le plus régulier, …)
  + Système de trophées (ex 100 dessins bien faits = trophée argent)
  + Défi journalier sur un dessin « difficile » à prédire
* Compte utilisateur
  + Informations générales (Nom d’utilisateur, description optionnelle, mot de passe)
  + Accès profil (trophées obtenus, points, voir les classements)

**Parcours utilisateur type** :

* L’utilisateur se connecte sur notre site et se crée un compte (pas besoin du mail pour que les enfants puissent y jouer facilement), s’il n’est pas connecté il peut tout de même consulter les règles et jouer au jeu (avec un peu moins de fonctionnalités)
* Une fois connecté, il arrive sur une page d’accueil, sur cette page il pourra consulter le classement général ainsi que les récompenses qu’il a obtenues. De plus, il aura accès à différentes pages
  + Une page « notez » ou il pourra noter les dessins des autres joueurs
  + Une page « profil » où il pourra modifier son pseudo et son mot de passe
  + Une page « jouer » qui lui donnera accès à notre jeu
* L’utilisateur lance le jeu
  + Il entre alors le nom de ce qu’il veut dessiner (girafe, voiture, …) et le temps qu’il souhaite avoir à disposition
  + Il peut ensuite dessiner dans une case prévue à cet effet.
  + Quand il a terminé, la prédiction s’affiche
  + Des points / récompenses lui sont attribués s’il a « bien dessiné » et un bouton lui permet de partager son dessin à la communauté

**Objectifs du projet :**

* Proposer une expérience de jeu assez innovante aux enfants
* Améliorer notre modèle de machine learning en alimentant notre jeu de données avec des dessins faits par des enfants
* Se servir de ce projet comme tremplin pour développer des applications futures qui pourraient intéresser des investisseurs

**Utilisateurs potentiels :**

* Les enfants, principalement entre 6 et 15 ans, à peu près 85 % de nos joueurs selon nous
* Les autres utilisateurs (adolescents, personnes âgées, …)

**Particularités techniques :**

* Utilisation du deep learning au sein de l’application (interaction entre notre modèle et l’application, besoin de vitesse pour que notre utilisateur puisse avoir le résultat à temps, …)
* Besoin de récupérer les dessins faits par l’utilisateur dans le cas où la prédiction était « presque bonne »

**Coût du développement de l’application :**

D’après notre équipe, le coût de développement serait composé principalement des salaires des développeurs.

Nous avons estimé que ce projet demanderait environ 4 mois de développement pour 3 développeurs en sortie d’école, chacun payé 3200 euros par mois.

Ce qui porterait le coût de l’application à environ 29 000 euros si nous rajoutons quelques coûts annexes (*exemple* : l’hébergement de notre application web).

# II - Spécifications générales et techniques

## **Périmètre fonctionnel**

**Objectif** **Acteur** **Impact**   **Fonctionnalité**

Règles du jeu avec photos explicatives

S’informer sur le jeu

Nouveau visiteur anonyme

Jeu de dessin

Tester le jeu (version basique)

Formulaire d’inscription

Créer un compte utilisateur

Amuser les joueurs Entrainer nos modèles

Système de trophées

Jouer (version complète)

Noter les dessins des autres pour avoir des points

Utilisateur inscrit

Avoir envie de rejouer (long therme)

Système de classement

Modifier ses informations

Profil utilisateur complet (modifiable)

Intégrer liens partage twitter et facebook

Administrateur

Back-office

Recommander le jeu

Gestion des bases de données

Créer et entrainer des modèles

Afficher KPIs du site

Cette partie ne sera pas implémentée dans la première version du produit livrée mais sera facilement ajoutable à l’avenir

## **Arborescence du site**

### **Arbo Utilisateur classique**

**Niveau 1**

**Niveau 0**

**Niveau 2**

PAGE D’ACCUEIL

Règles

Inscription / Connexion

Jeu (version basique)

Accueil simple

Accueil amélioré

Notation des dessins

Profil utilisateur

Trophées, classement, …

### **Arbo Utilisateur administrateur**

**Niveau 1**

**Niveau 0**

**Niveau 2**

PAGE D’ACCUEIL

Inscription / Connexion

Page d’accueil Admin

Créer/entrainer des modèles

Gestion des BDDs

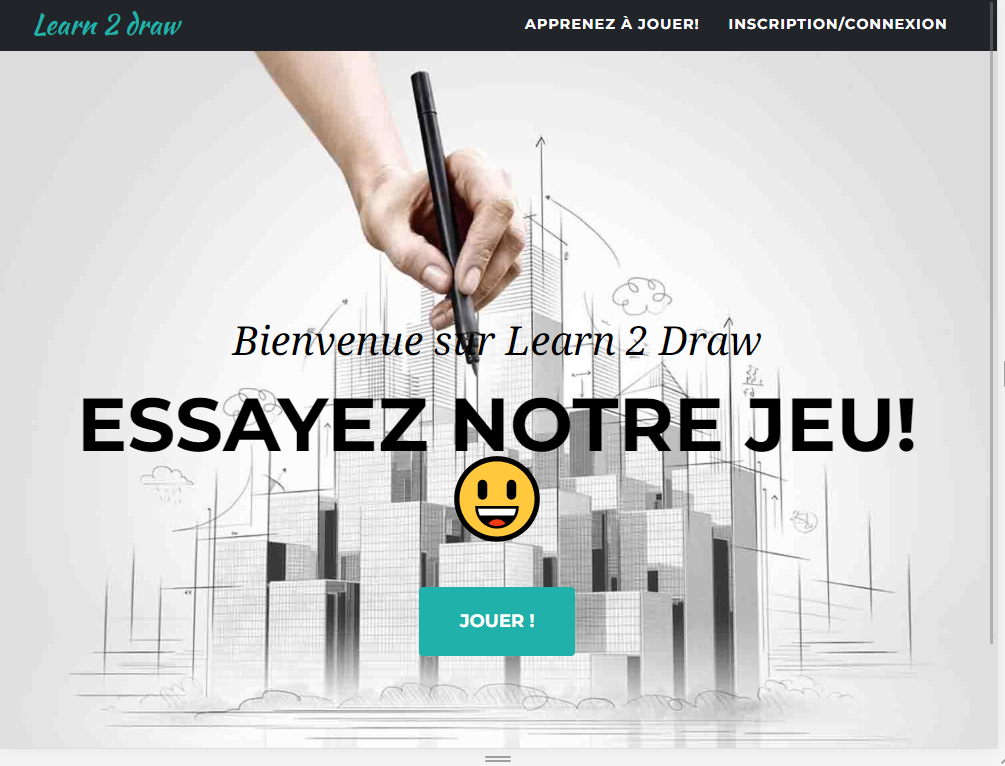
Consulter les statistiques

**Niveau 3**

## **Fonctionnalités pour chaque page**

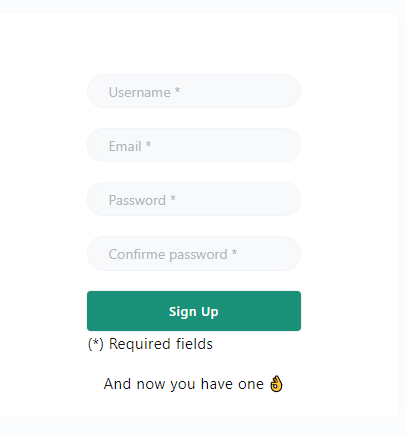
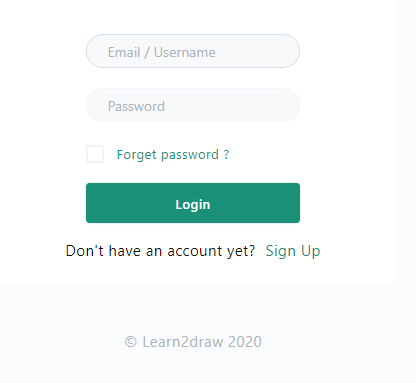
### **Frontend**

#### **Page landing.html**



|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Page d’accueil basique du site | La page d’accueil du site, elle permet d’accéder aux différentes pages de niveau 1 (Inscription / connexion, règles, jeu basique) |  |

**Page login.html** (+ signup)



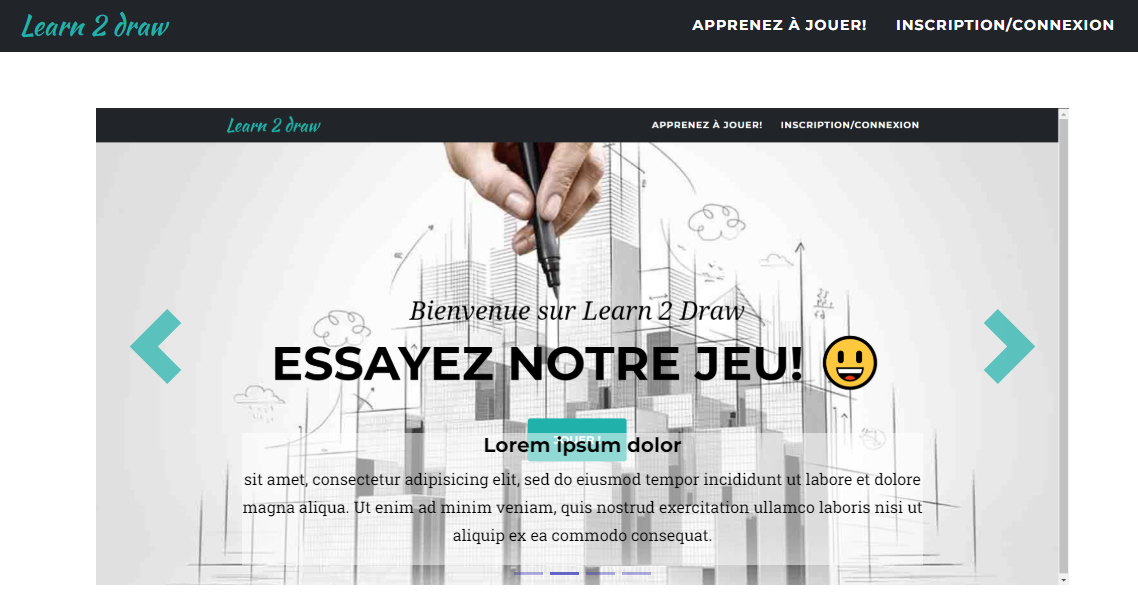
Inscription

Connexion

|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| S’inscrire en tant que nouvel utilisateur | L’utilisateur final devra entrer diverses informations pour obtenir un compte | \*Vérifications des informations entrées |
| Se connecter à l’application web | L’utilisateur final devra entrer son nom d’utilisateur et son mot de passe dans le formulaire | \*Vérifications des informations entrées  \*S’assurer de l’unicité d’une session utilisateur (Token) |

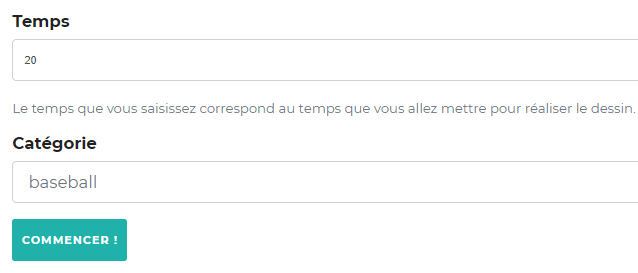
#### 

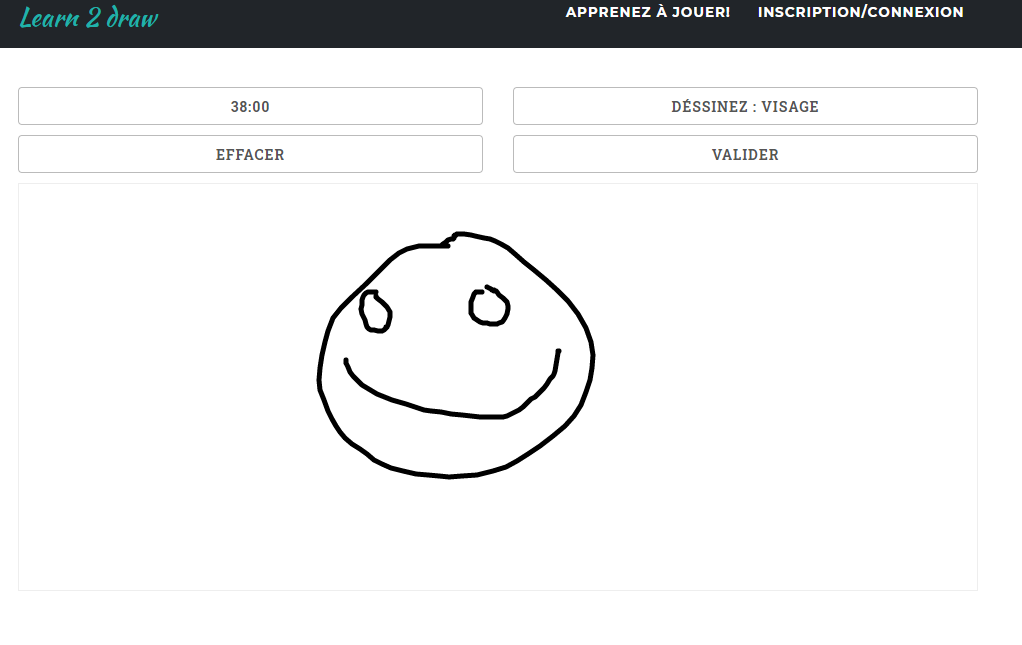
#### **Page rules.html**

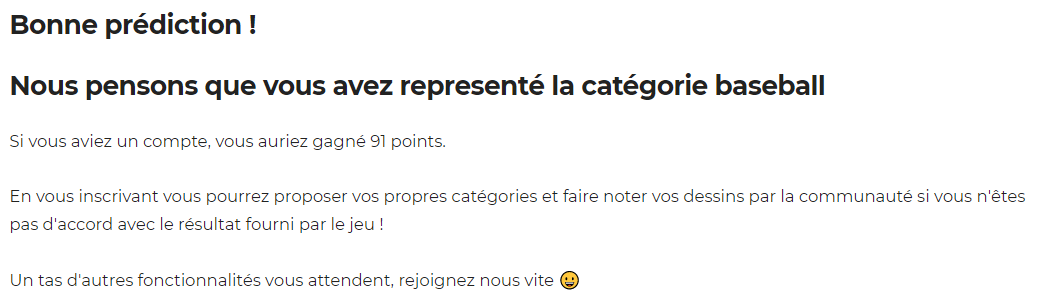


|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Afficher l’ensemble des règles | Des règles s’affichent dans un carrousel (ensemble de photos qui défilent) |  |

#### **Page play.html**

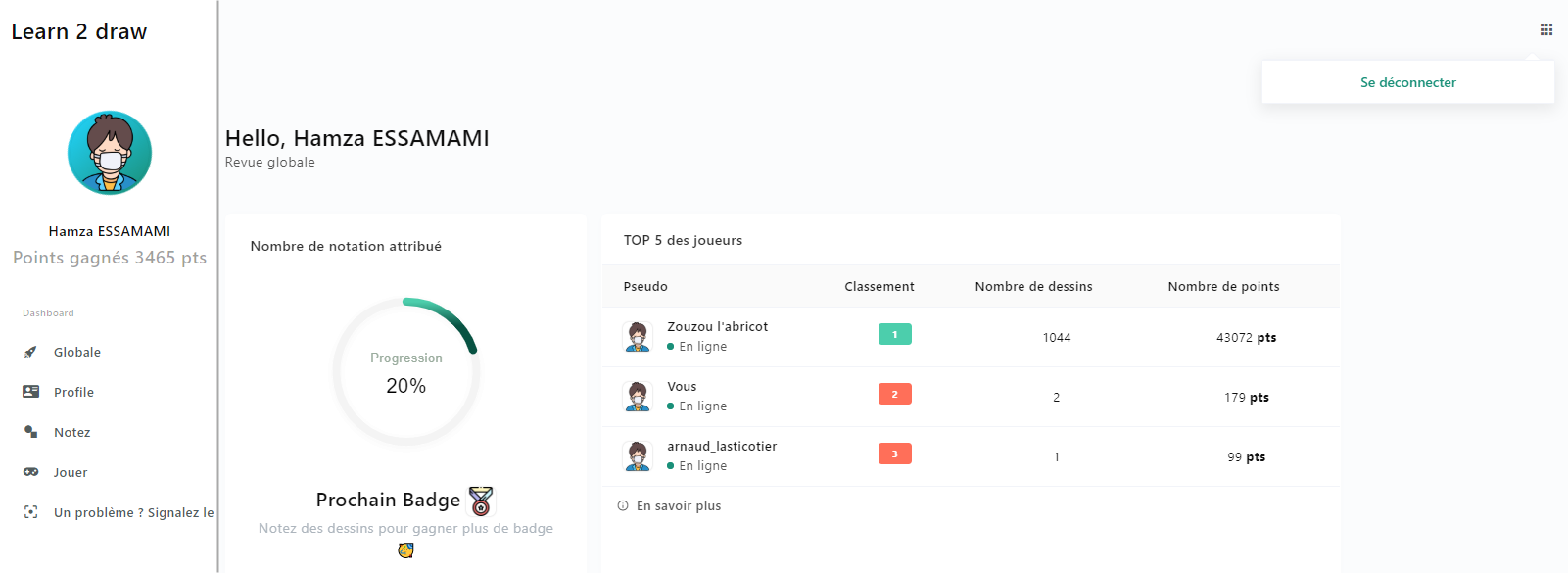






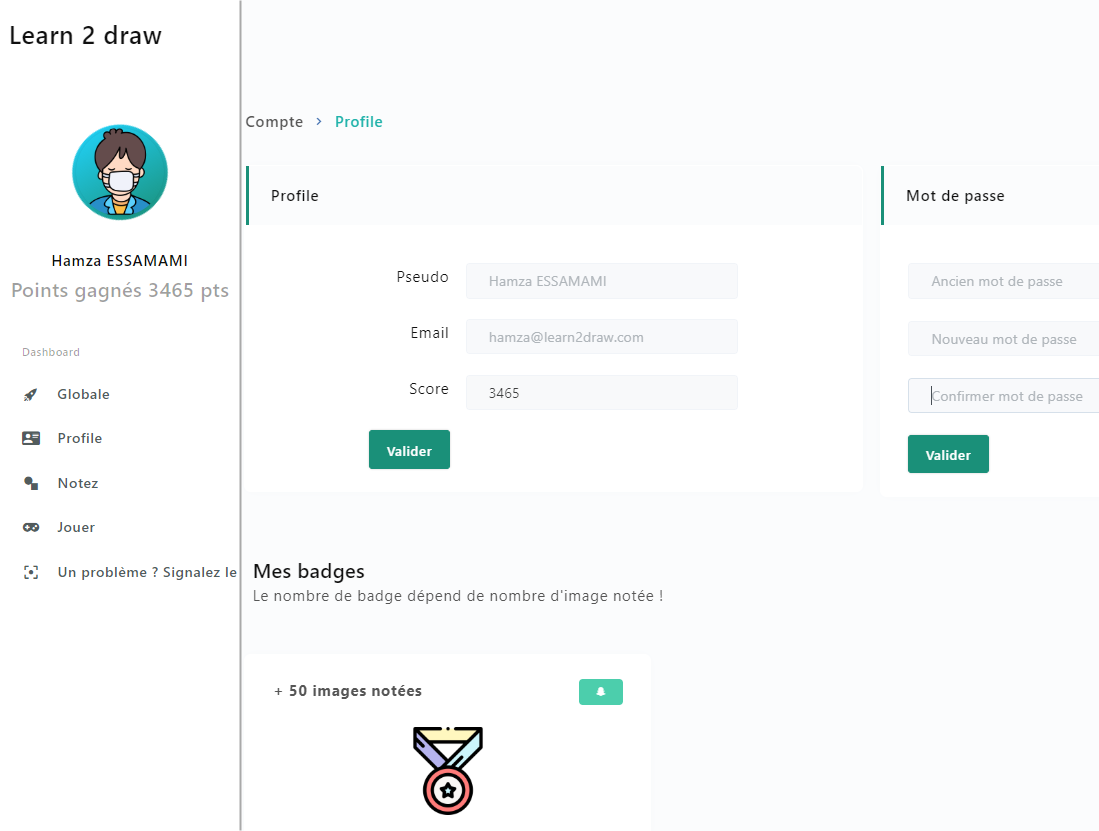
|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Dessiner dans une fenêtre centrale (selon les critères choisis) | Après avoir choisi une catégorie et le nombre de secondes disponibles dans un formulaire, dessiner dans une fenêtre au centre de l’écran dans le temps imparti  (L’utilisateur peut effacer son dessin avant la fin du temps imparti) | \*Eviter les latences pour avoir la meilleure expérience de jeu possible pour l’utilisateur |
| Afficher la prédiction faîte par notre algorithme | A la fin du temps imparti, notre algorithme propose une catégorie pour le dessin (chien par exemple) | \* Avoir les meilleures prédictions possibles, même quand notre modèle devra gérer beaucoup de catégories |
| Partager le dessin pour le faire noter par la communauté | L’utilisateur peut choisir de soumettre le dessin à la communauté, il sera alors noté plus par d’autres utilisateurs (si les utilisateurs le notent bien, il sera récupéré pour les futurs entrainements de nos modèles de machine learning) |  |
| Sauvegarder le dessin | Le dessin sera sauvegardé dans un dossier mais inaccessible pour l’utilisateur |  |

#### **Page home\_user\_app.html**



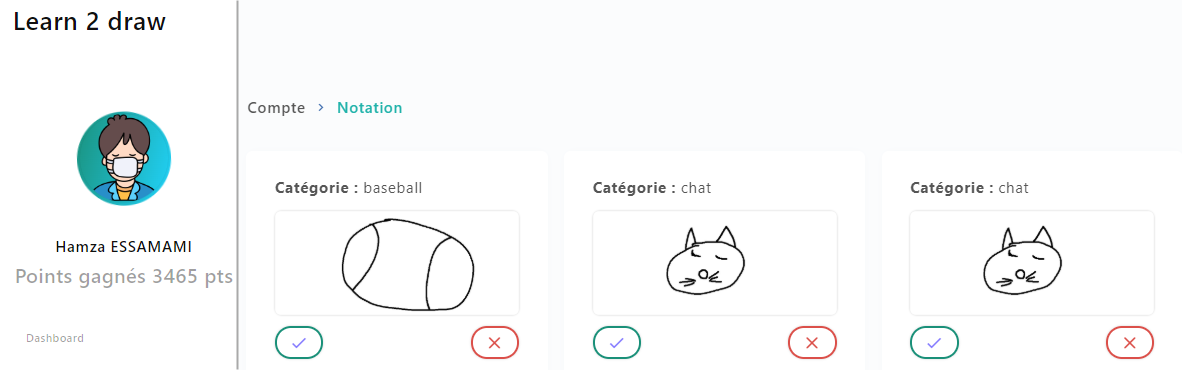
|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Consulter quelques informations sur l’utilisateur courant (nom, nombre de points actuels, progression du badge (cuivre, argent, …)) | L’utilisateur connecté peut consulter quelques informations sur son profil, les autres informations se trouveront sur la page profil |  |
| Accéder au classement des joueurs | Le classement des meilleurs joueurs s’affiche sur la gauche |  |

#### **Page profile\_user\_app.html**



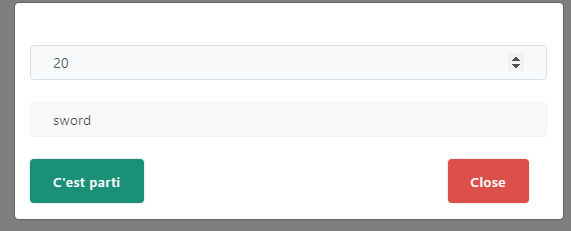
|  |  |  |
| --- | --- | --- |
| Modifier les informations utilisateurs (mail, nom d’utilisateur, mot de passe) | L’utilisateur peut modifier les informations de son profil quand il le souhaite (mail, nom d’utilisateur, mot de passe) | \*Vérifications des informations entrées |
| Consulter les différents badges obtenus | L’utilisateur peut consulter les badges qu’il a obtenu, avoir des indices pour savoir comment obtenir les suivants (sauf si badge « caché ») |  |

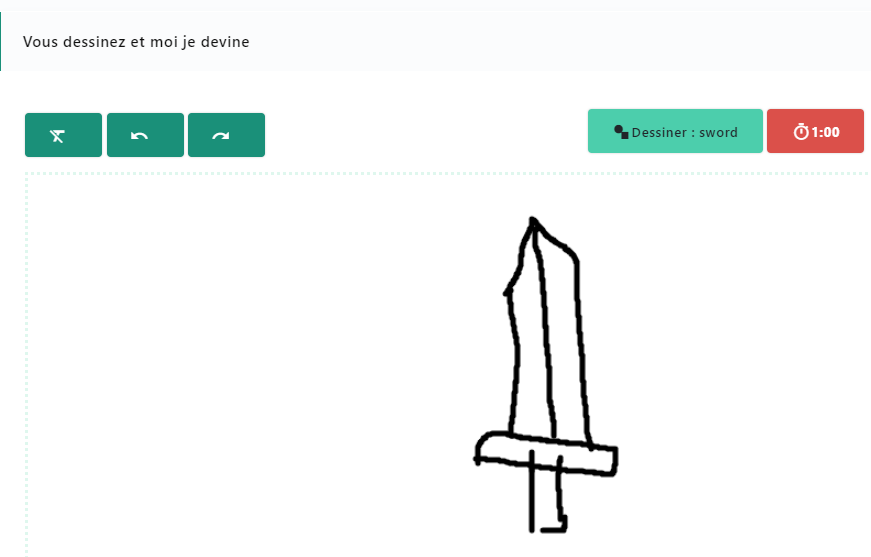
#### **Page notation\_user\_app.html**



|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Noter un dessin | De nombreux dessins apparaissent (en priorité les dessins « vieux » avec peu de votes), l’utilisateur peut choisir un de ces dessins et indique s’il pense qu’il représente bien la catégorie mentionnée, s’il a répondu la même chose que la plupart des utilisateurs, il sera récompensé avec des points | \*Un utilisateur ne doit pas pouvoir noter ses propres dessins ni ceux qu’il a déjà  noté |
| Valider une image | Si une image a été notée 100 fois, un score lui est attribuée et elle est validée (changement de statut) si elle obtient plus de 90% de votes « yes »  (Quand une image a été notée 100 fois, nous envisageons de supprimer ces données de la base pour libérer de l’espace) |  |
| Créer un nouveau dataset | Quand il y a 1000 images d’une catégorie qui sont validées, un nouveau dataset est créé (s’il en existe déjà un, fusionne le nouveau avec le récent)  Change également le statut des images (du statut « notées » à « validées / ou non »)  Attribution de points aux joueurs qui ont votés comme la majorité |  |

### **Page game\_user\_app.html**



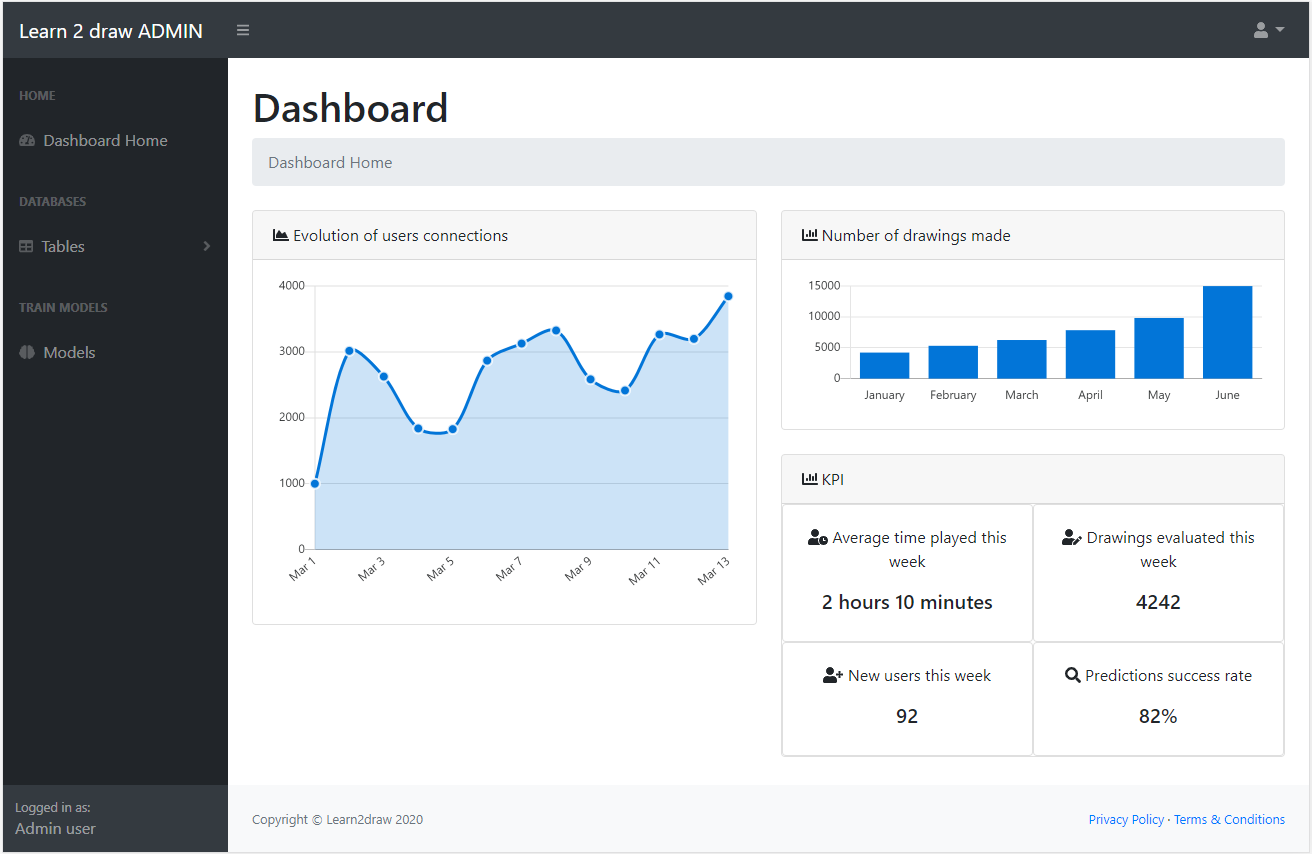




|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Dessiner dans une fenêtre centrale (selon les critères choisis) | Après avoir choisi une catégorie (entrée libre contrairement au jeu basique) et un temps en secondes, dessiner dans une fenêtre au centre de l’écran dans le temps imparti  (L’utilisateur peut reculer ou avancer d’un trait et effacer son dessin) | \*Eviter les latences pour avoir la meilleure expérience de jeu possible pour l’utilisateur |
| Afficher la prédiction faîte par notre algorithme | A la fin du temps imparti ou lors du clic sur le bouton « Valider », notre algorithme propose une catégorie pour le dessin (chien par exemple)  Si la catégorie n’est pas gérée par notre modèle et n’est pas dans notre base de données, elle sera créée. |  |
| Partager le dessin pour le faire noter par la communauté | L’utilisateur peut choisir de soumettre le dessin à la communauté, il sera alors noté plus par d’autres utilisateurs (si les utilisateurs le notent bien, il sera récupéré pour les futurs entrainements de nos modèles de machine learning) |  |
| Sauvegarder le dessin | Les dessins sont sauvegardés dans des dossiers et dans notre base de données |  |

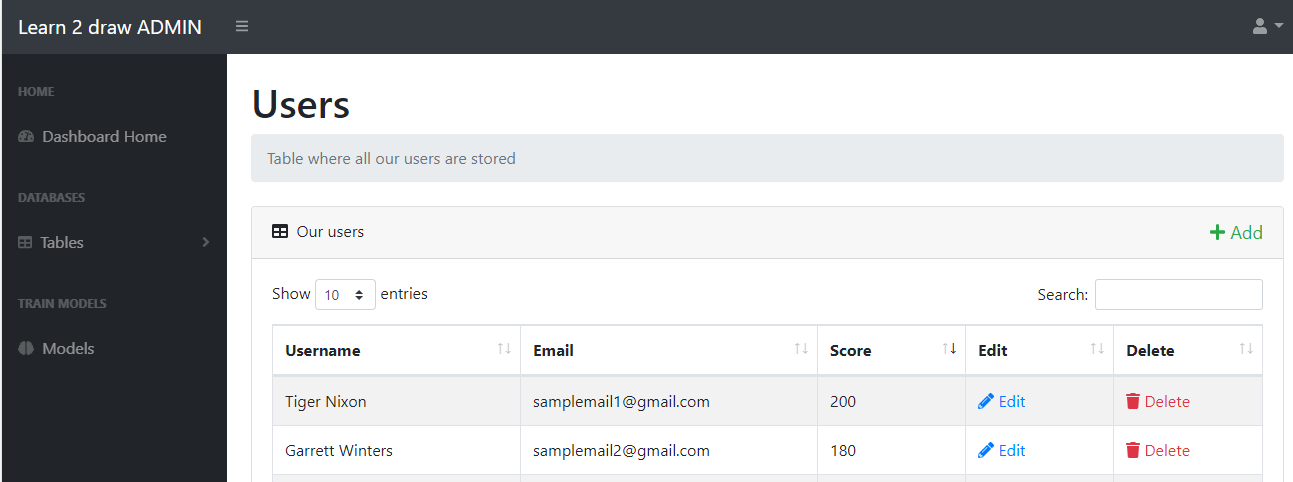
### **Back Office**

#### **Page admin\_home.html**



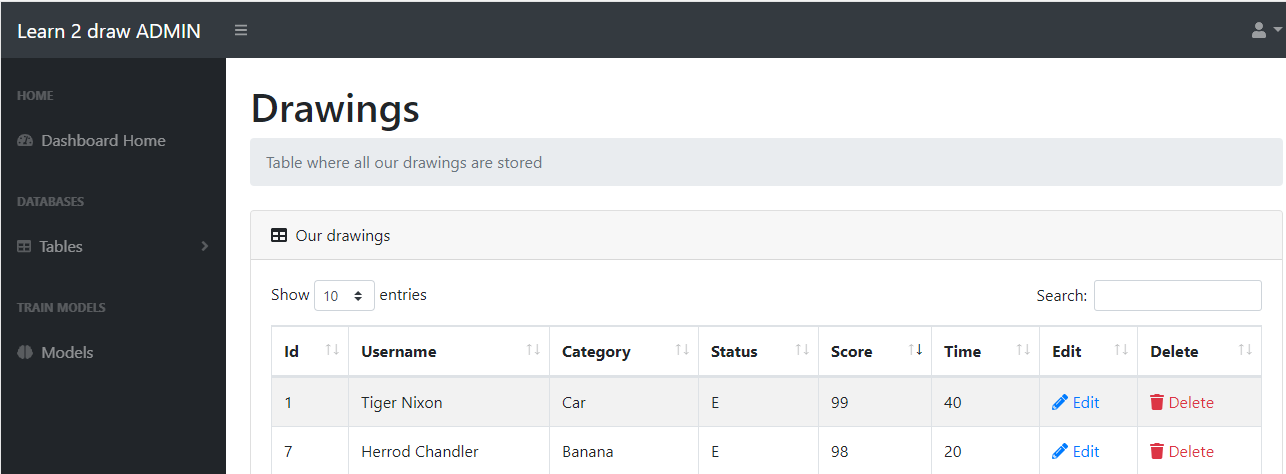
|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Consulter les KPI du site | Lorsqu’un administrateur arrive sur la page, il peut directement voir les informations les plus importantes sur « l’état de santé » du site (suivi de la popularité, nombre de parties jouées, …) et ce sur plusieurs périodes grâce à l’utilisation de filtres |  |

#### **Page manage\_users.html**



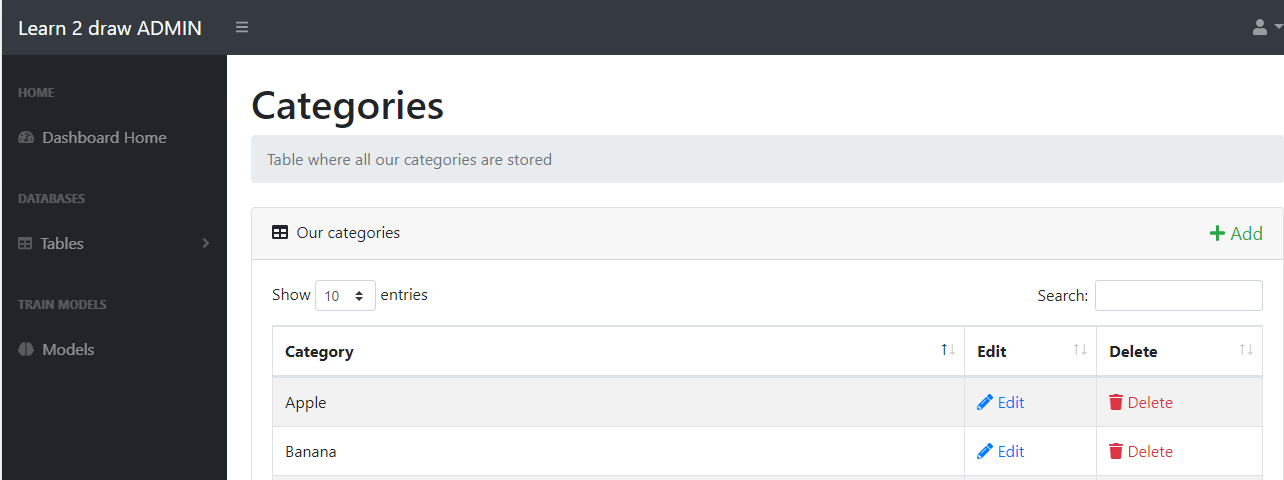
|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Créer un nouvel utilisateur | L’administrateur peut créer un nouvel utilisateur en complétant un formulaire (comme le formulaire d’inscription classique) | \*Vérifications des informations entrées (Nom d’utilisateur d’une bonne taille et pas déjà présent dans la BDD, …) |
| Lister l’ensemble des utilisateurs | Tous les utilisateurs s’affichent clairement sur la page, il est facile de retrouver un utilisateur précis à l’aide d’une barre de recherche et de filtres |  |
| Modifier les informations d’un utilisateur | L’administrateur peut choisir un utilisateur précis et modifier les informations le concernant à l’aide d’un formulaire | \*Vérifications des informations entrées (Nom d’utilisateur d’une bonne taille et pas déjà présent dans la BDD, …) |
| Supprimer un utilisateur | Bouton de suppression pour chaque utilisateur avec demande de confirmation pour éviter les suppressions involontaires |  |

#### **Page manage\_drawings.html**



|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Lister l’ensemble des dessins réalisés (au moins les X derniers) | Tous les dessins s’affichent clairement sur la page, il est possible de retrouver un dessin précis à l’aide d’une barre de recherche et de filtres |  |
| Modifier les informations d’un dessin | L’administrateur peut changer le statut du dessin (pour l’instant pas d’autres modifications retenues) | \*Vérifications des informations entrées |
| Supprimer un dessin | Bouton de suppression pour chaque dessin avec demande de confirmation pour éviter les suppressions involontaires |  |

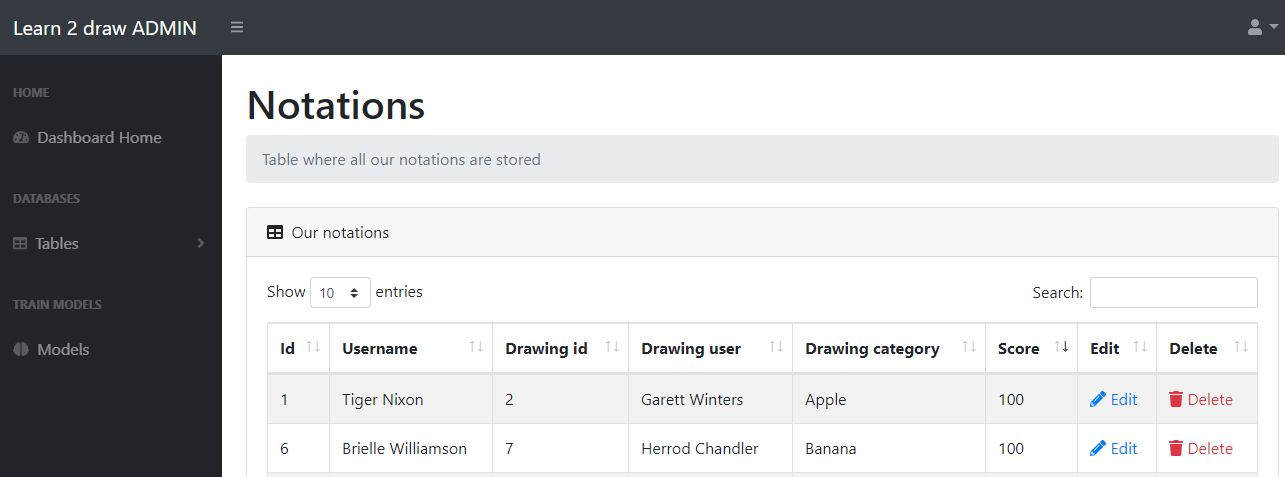
#### **Page manage\_categories.html**



|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Créer une nouvelle catégorie | L’administrateur peut créer une nouvelle catégorie en complétant un formulaire (simple input suffisant) | \*Vérifications des informations entrées (pas de catégories avec le même nom) |
| Lister l’ensemble des catégories | Toutes les catégories s’affichent clairement sur la page, il est facile de retrouver une catégorie précise à l’aide d’une barre de recherche et de filtres |  |
| Modifier les informations d’une catégorie | L’administrateur peut modifier le nom d’une catégorie (pour l’instant seule option envisagée) | \*Vérifications des informations entrées (Pas de catégories avec le même nom) |
| Supprimer une catégorie | Bouton de suppression pour chaque catégorie avec demande de confirmation pour éviter les suppressions involontaires | \* Supprimer une catégorie peut être dangereux pour le modèle actuellement utilisé, s’il utilise la catégorie supprimée il sera recréé sans cette dernière  (de plus il est impossible de supprimer / modifier les 2 catégories initiales de notre appli : broom et baseball) |

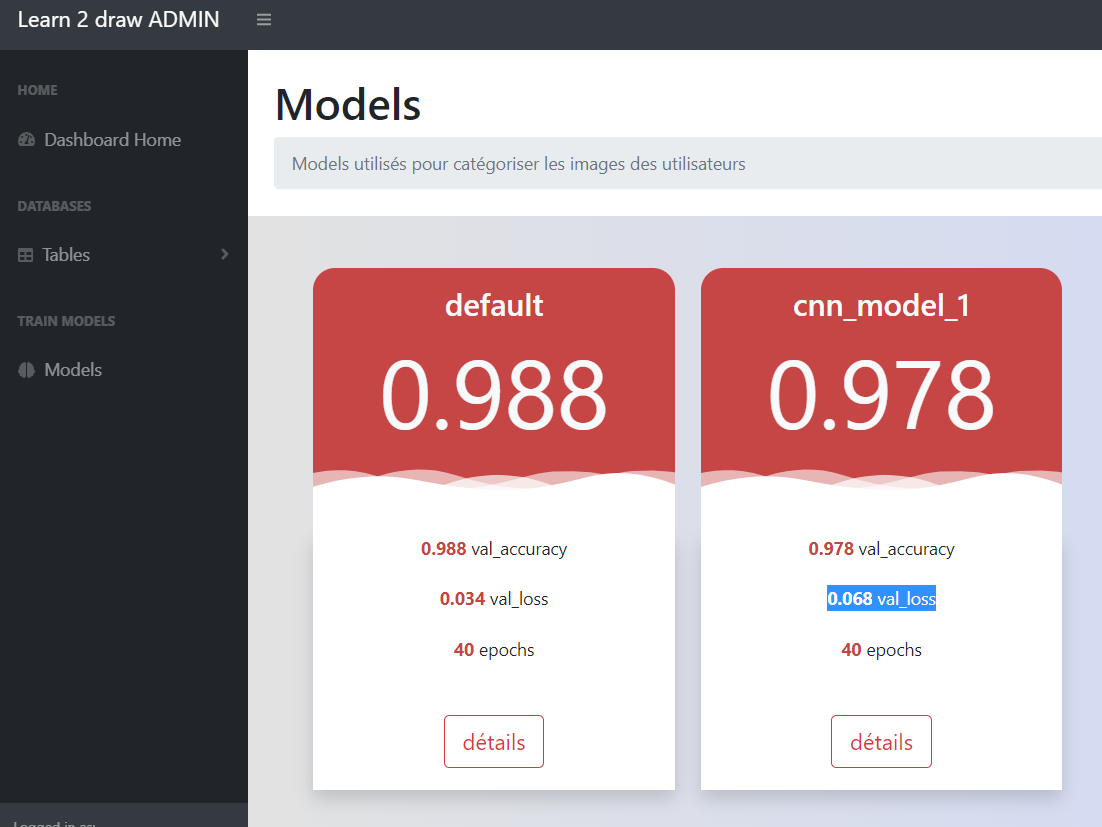
### 

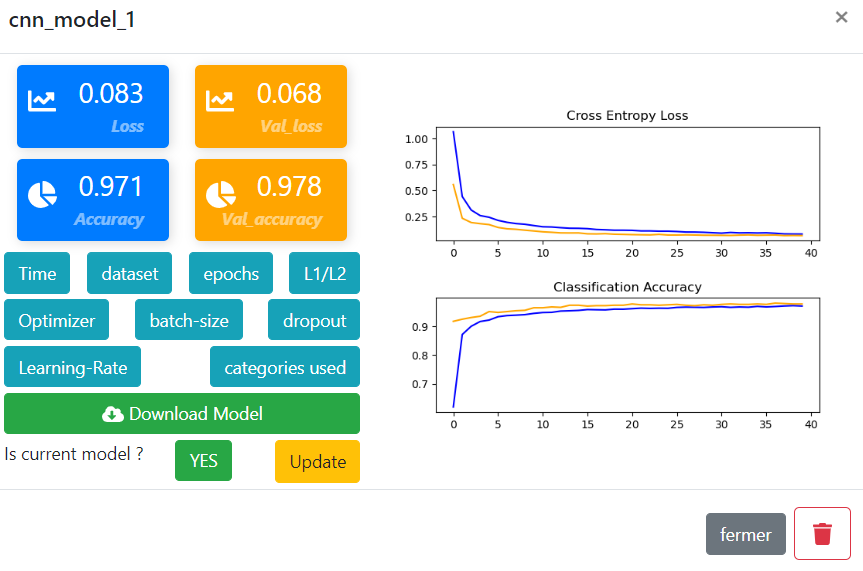
#### **Page manage\_notations.html**



|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Lister l’ensemble des notes attribuées aux dessins | Toutes les notes s’affichent clairement sur la page, il est possible de retrouver une note précise à l’aide d’une barre de recherche et de filtres |  |
| Modifier les informations d’une note | L’administrateur peut changer le score d’une note (pour l’instant seule option envisagée) |  |
| Supprimer une note | Bouton de suppression pour chaque note avec demande de confirmation pour éviter les suppressions involontaires |  |

#### **Page manage\_models.html**





|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Description** | **Contraintes potentielles** |
| Lister l’ensemble des modèles existants et afficher leurs résultats | Les différents modèles s’affichent sur la page, aperçu de leurs performances en un coup d’œil avec la val\_accuracy et la val\_loss |  |
| Modifier les paramètres des modèles | L’administrateur peut changer les hyperparamètres des modèles puis entrainer à nouveau un modèle (ouverture d’un modal puis modification d’un formulaire) | \*Vérification des informations entrées |
| Créer un nouveau modèle | L’administrateur peut créer un nouveau modèle en complétant un formulaire | \* Vérification des informations entrées |
| Supprimer des modèles | Bouton de suppression avec confirmation utilisateur pour éviter les actions involontaires |  |
| Télécharger la structure du modèle actuel | Bouton download model pour récupérer la structure d’un modèle |  |
| Sélectionner un modèle en tant que « modèle actuel » | Bouton pour changer le modèle actuellement utilisé pour faire des prédictions (YES quand le modèle consulté est utilisé, sauf pour default qui est toujours valide (modèle de secours)) |  |

# III - Architecture technique

u d’apprentissage

**Schéma architecture technique**

**Utilisateur**

**Serveur Web**





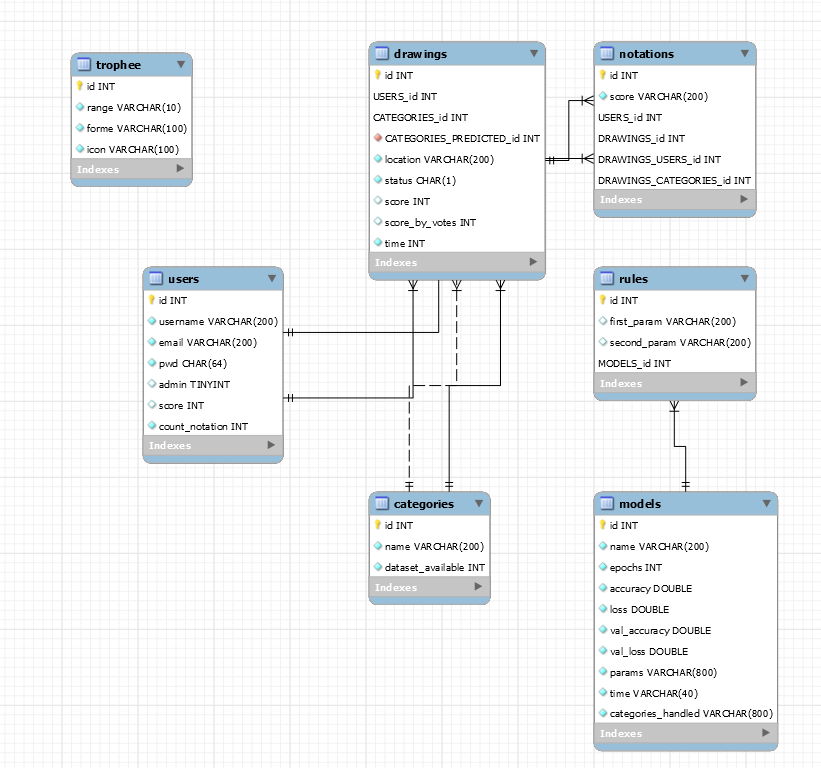
**FrontEnd**



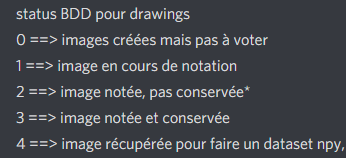
**Serveur de données**

**BackEnd**



Explications de certains champs : **Drawings => status** : Nombre qui permet de savoir si une image est en cours de notation, validée par les utilisateurs etc.



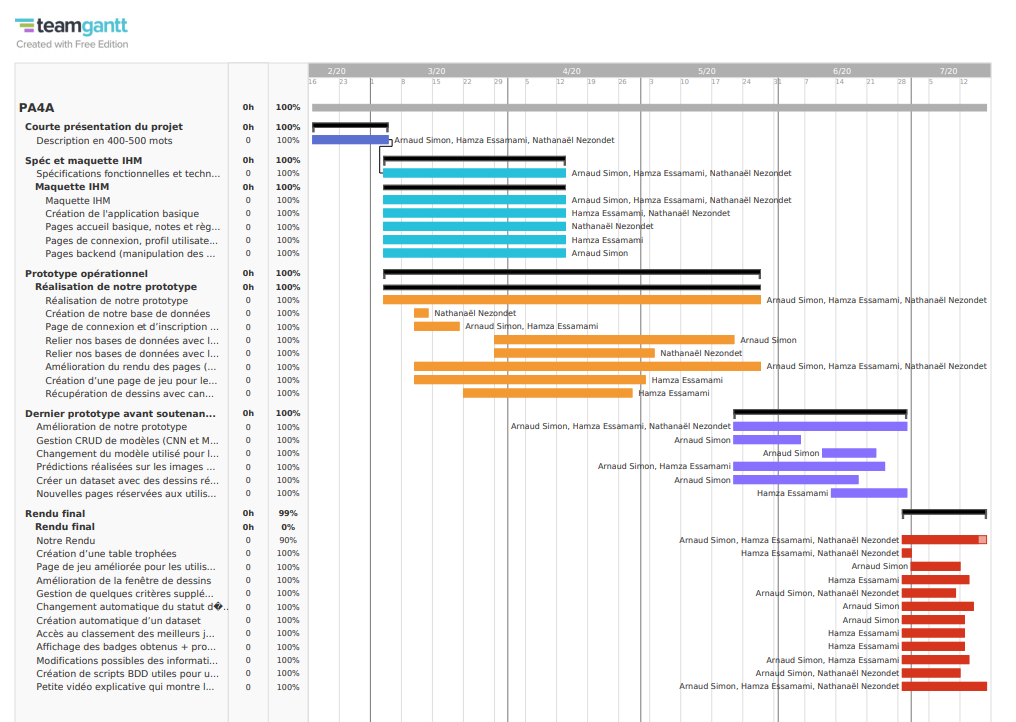
**Catégories => dataset\_available** : permet de savoir si cette catégorie devra être checkée par le programme pour tenter un entrainement

## **Technologies choisies**

|  |  |  |
| --- | --- | --- |
| **Technologie retenue** | **Description** | **Besoins remplis** |
|  | Outil opensource d’apprentissage automatique développé par Google.  Il est utilisé dans de nombreux projets de machine learning | \*Entrainer nos modèles de machine learning |
|  | Framework opensource de développement web en python, il permet de concevoir plus rapidement des sites web et d’y intégrer facilement la programmation python | \*Utiliser facilement du python dans notre application web |
|  | SQLAlchemy est un toolkkit open source SQL et un ORM écrit en python. Il permet de se connecter aux sources de données facilement et d’effectuer des requêtes sur ces dernières | \*Se connecter à nos bases de données MySQL |
|  | Bibliothèque python qui permet la manipulation et l’analyse de données. Elle propose des strcutures de donnés très utiles comme les Dataframes et peut manipuler aisément des tableaux numériques ainsi que des séries temporelles | \*Placer nos données dans des structures facilement manipulables  \*Effectuer des requêtes SQL en utilisant une connexion fournie par SQLAlchemy |
|  | PyMySQL est une Bibliothèque python qui permet de se connecter et d’effectuer des requêtes sur des bases de données MySQL | \*Interagir avec nos bases de données côté Python |
|  | phpMyAdmin est une application web de gestion pour les SGBD MySQL écrite principalement en PHP | \*Stock de nos bases de données MySQL  [Gestion simplifiée par rapport à un serveur MySQL classique ?] |

# IV - Planning et répartition des tâches

<https://prod.teamgantt.com/gantt/schedule/?ids=2042178#&ids=2042178&user=&custom=&company=&hide_completed=false&date_filter=&color_filter=>



# V - Travail réalisé (tableau des tâches) et difficultés rencontrées

**Tableaux des tâches**

|  |  |
| --- | --- |
| **Tâches** | **Noms** |
| **I - Description du projet en 400-500 mots** |  |
| **Description du projet en 400-500 mots**  *Description* : Petite description du projet qui rassemble la plupart des informations générales sur notre application (fonctionnalités principales, utilisateurs potentiels, parcours utilisateur type, objectif du projet, ...) | Arnaud, Hamza, Nathanaël |
| **II - Spécifications et maquette IHM** |  |
| **Spécifications fonctionnelles et techniques**  *Description* : Documentation qui regroupe l’ensemble des technologies utilisées dans notre projet (Flask, SQLAlchemy, Pandas, ...), le périmètre fonctionnel et l’architecture globale de notre application, ainsi qu’une description exhaustive des fonctionnalités de chaque page. | Arnaud, Hamza, Nathanaël |
| **Maquette IHM** |  |
| **Création de l’application basique**  *Description* : Création des premiers templates Jinja, définition des bibliothèques à installer en priorité, ... | Nathanaël, Hamza |
| **Pages accueil basique, notes et règles**  *Description* : Réalisation de templates de ces pages (HTML, CSS, JS) | Nathanaël |
| **Pages de connexion, profil utilisateur**  *Description* : Réalisation de templates de ces pages (HTML, CSS, JS) | Hamza |
| **Pages backend (manipulation des bases de données)**  *Description* : Réalisation de templates de ces pages (HTML, CSS, JS) | Arnaud |
| **III - Prototype opérationnel** |  |
| **Création de notre base de données**  *Description* : Réalisation d’un schéma pour notre base de données | Nathanaël |
| **Page de connexion et d’inscription fonctionnelles**  *Description* : Relier ces pages à notre base de données, gérer les cas spéciaux (si utilisateur déjà créé, empêcher la création, …) | Hamza, Arnaud |
| **Relier nos bases de données avec les opérations Backend** (créer, lister, modifier et supprimer des enregistrements)  *Description* : Relier ces pages à notre base de données en tentant de régler un maximum de cas spéciaux (les cas spéciaux qui étaient vraiment trop nuisibles pour l’application ont étés interdits) | Arnaud |
| **Relier nos bases de données avec la page « notation »** (noter des dessins et enregistrer ces notes)  *Description* : Affichage des données de notre BDD (dessins créés par nous-même en “dur” dans un premier temps) et possibilité de leur attribuer une note (ressemblance ? ==> oui ou non) | Nathanaël |
| **Amélioration du rendu des pages** (HTML, CSS, JS )  *Description* : Rajout de quelques éléments graphiques intéressants, modaux bootstrap stylisés, …) | Arnaud, Hamza, Nathanaël |
| **Création d’une page de jeu pour les utilisateurs non connectés**  *Description* : Page de jeu simple où les utilisateurs peuvent dessiner, choix du nombre de secondes disponibles ainsi que de la catégorie à dessiner | Hamza |
| **Récupération de dessins avec canvas dans des dossiers**  *Description* : Quand les dessins sont créés par les utilisateurs, ils sont sauvegardés dans des dossiers (pas en base comme ce mode de jeu est réservé aux utilisateurs non connectés) | Hamza |
| **IV – Dernier prototype avant la soutenance** |  |
| **Gestion CRUD de modèles (CNN et MLP)**  *Description* : Affichage des différents modèles existants, possibilité d’en créer de nouveaux avec quelques hyperparamètres (batchsize, epochs, optimizer, …), modification (suivi d’un nouvel entrainement) et suppression possible | Arnaud, Nathanaël |
| **Changement du modèle utilisé pour les prédictions**  *Description* : Possibilité de sélectionner le modèle qui va être utilisé pour prédire les catégories des dessins | Arnaud |
| **Prédictions réalisées sur les images dessinées par les utilisateurs**  *Description* : Récupération des dessins utilisateurs, multiples conversions (redimensionnement, suppression du filtre de transparence et du channel “Alpha”, remplacement de certaines valeurs pour obtenir une image utilisable par notre modèle, …)  Retourne un résultat à l’utilisateur qui sait s’il a bien dessiné ou non ainsi que le nombre de points qu’il a reçu | Arnaud |
| **Créer un dataset avec des dessins réalisés par les utilisateurs**  *Description* : Petite simulation pour créer un dataset.npy composé de dessins réalisés par des utilisateurs, dataset ensuite utilisé par un modèle s’il est mis à jour / créé (cherche les dernières catégories existantes en BDD avec un fichier npy disponible) | Arnaud |
| **Nouvelles pages réservées aux utilisateurs connectés** (début de construction des pages)  *Description* : Templates et quelques interactions BDD réalisés pour les pages réservées aux utilisateurs | Hamza |
| **V – Résultat final** |  |
| **Création d’une table badges**  *Description* : Création d’une table badges qui recense l’ensemble des trophées obtenables par les utilisateurs | Nathanaël, Hamza |
| **Page de jeu améliorée pour les utilisateurs connectés**  *Description* : Fonctionnalités supplémentaires : réel choix d’une catégorie qui est créée si non existante en BDD et possibilité de soumettre son dessin au vote) | Arnaud |
| **Amélioration de la fenêtre de dessins**  *Description* : Possibilité de faire une sorte de « marche avant » / « marche arrière » sur notre fenêtre de dessin | Hamza |
| **Gestion de quelques critères supplémentaires dans la création d’un modèle**  *Description* : Rajout de quelques hyperparamètres, renvoi d’informations complémentaires sur notre modèle (catégories gérées par le modèle par exemple) | Arnaud, Nathanaël |
| **Changement automatique du statut d’un dessin noté**  *Description* : Quand 100 notes sur un dessin, changement de statut (image utilisable pour créer un dataset ou non + distribution de points pour les utilisateurs ayant voté comme la majorité) | Arnaud |
| **Création automatique d’un dataset composé des dessins des utilisateurs**  *Description* : Quand 1000 dessins sont validés par les votes (statut == 3), créer un nouveau dataset.npy avec ces images (fusion si déjà existant), change ensuite le statut des dessins (sauvegardés dans dataset, statut == 4) | Arnaud |
| **Accès au classement des meilleurs joueurs (top 5)**  *Description* : Affichage des meilleurs joueurs sur l’écran d’accueil des utilisateurs connectés | Hamza |
| **Affichage des badges obtenus par un utilisateur et la progression actuelle pour atteindre le suivant**  *Description* : Affichage des informations sur les badges (récupération des informations en base et traitements supplémentaires pour obtenir un bel affichage) | Hamza |
| **Modifications possibles des informations « profil » par un utilisateur** (changement de mot de passe, nom d’utilisateur ou email)  *Description* : Création de formulaires pour que les utilisateurs puissent modifier leurs informations en base, implémentation de vérifications pour éviter les erreurs en BDD (utilisateur déjà existant, ...) | Arnaud, Hamza |
| **Création de scripts BDD utiles pour une démo**  *Description* : Créations de quelques scripts SQL pour créer beaucoup de dessins, les insérer en BDD, créer des notes, ...  pour montrer ce qu’il se passe quand 1000 dessins sont validés / quand une image a reçu 100 votes)  (finalement insertion des exemples nécessaires dans le fichier de création de la base (learn2draw\_db.sql) | Arnaud, Nathanaël |
| **Petite vidéo explicative qui montre les cas d’usages principaux de notre application**  *Description* : Petite vidéo retraçant les fonctionnalités principales de notre application (dessiner un dessin et obtenir une prédiction pour les 2 modes de jeu différents (connecté / non connecté), interaction avec nos tables avec le backend, création et sélection d’un modèle, distribution de points à la 100 ème note d’un dessin, création d’un dataset lorsque la 1000 ième image validée est insérée en base de données) | Arnaud, Nathanaël, Hamza |

**Difficultés rencontrées**

Au cours du développement de notre application, nous avons pu rencontrer certains problèmes qui nous ont fait malheureusement perdre du temps. Voici un petit récapitulatif des principales difficultés auxquelles nous avons dû faire face

### **Récupérer un dessin avec canvas**

Lorem ipsum HAMZA

### **Trouver des comment exploiter les données de quickdraw (traitement des données, création de modèles, …)**

Quick draw possède de nombreux datasets dans plusieurs formats différents

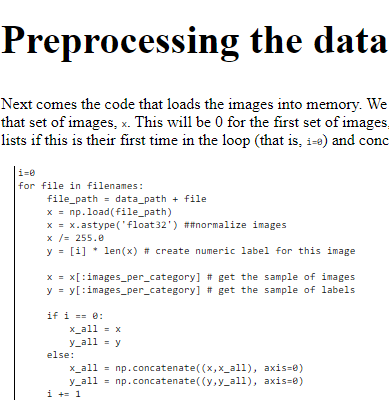
* .ndjson (grand JSON ou chaque objet est stocké sur une ligne)
* .bin (images stockées en binaire pour améliorer la compression et le chargement)
* .npy (images en niveau de gris stockées au format 28,28 dans des tableaux numpy)

Nous avons tout d’abord essayé d’exploiter les datasets au format .ndjson, mais certains problèmes sont apparus et les sources que nous avions trouvés n’étaient pas très explicites.

Après des recherches supplémentaires nous avons trouvé quelques ressources intéressantes qui expliquaient brièvement comment récupérer les fichiers .npy de quickdraw et les utiliser dans un modèle (notamment le livre Tensorflow 2.0 Quick Start Guide), il fût alors plus facile de créer un modèle utilisant ces fichiers .npy.

(site pour télécharger des datasets au format .npy <https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap?pli=1>)

**Extrait du livre**

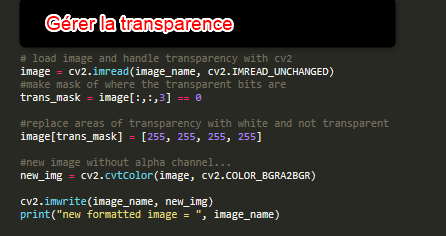


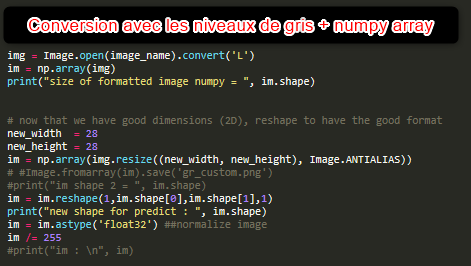
### **Transformer les dessins utilisateurs pour qu’ils puissent être utilisés par nos modèles (et donc trouver les bonnes catégories)**

Après avoir réussi à créer des modèles, il nous fallait pouvoir prédire les catégories des dessins des utilisateurs. Nous pensions au premier abord que la tâche allait être assez simple mais finalement de multiples opérations de conversion des images étaient nécessaires pour réaliser une prédiction.

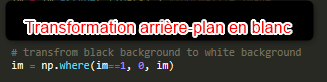
Après de multiples essais, nous avons trouvé une manière d’obtenir d’assez bonne prédictions à partir des modèles entrainés avec les données quickdraw (fichiers .npy), pour se faire nous avons utilisé la librairie de manipulation d’images open-cv.

Nous avons tout d’abord essayé de convertir l’image en niveau de gris mais nos prédictions étaient mauvaises, après quelques recherches sur le web nous sommes parvenus à la conclusion que le niveau de transparence de nos images « brouillait » les résultats.

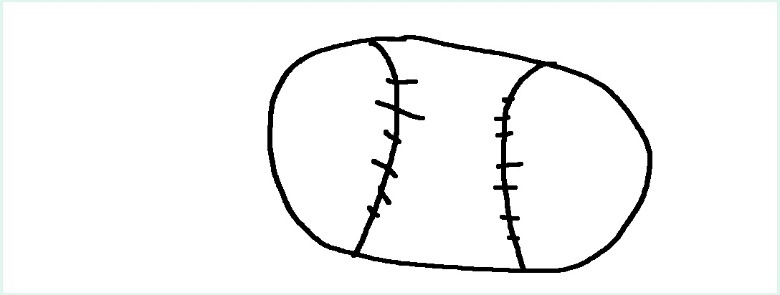
Grâce à open-cv, nous avons donc pu le retirer et convertir nos images en niveau de gris. 



Cependant, les prédictions étaient toujours mauvaises et nous avons également dû transformer l’arrière-plan des images (du noir vers le blanc)



Après ces modifications, les prédictions semblaient être sensiblement meilleures (**baseball à 93 %** pour l’image ci-dessous)



### **Créer des datasets au format .npy à partir des images des utilisateurs**

Après avoir réussi à prédire assez bien les catégories des datasets quickdraw, nous avons voulu essayer de créer des modèles utilisant les dessins des utilisateurs.

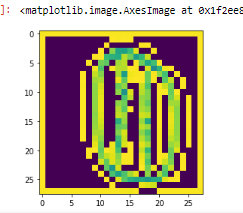
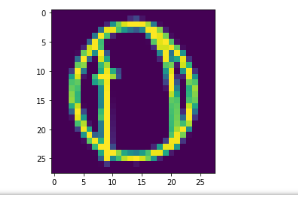
Pour ce faire, nous pensions qu’il fallait tout simplement traiter les images comme nous l’avions fait lors de la prédiction (sans les étapes de gestion de la transparence car lors de la prédiction les images sont resauvegardées sans transparence), mais étrangement les résultats furent totalement aberrants (prédiction de la nouvelle catégorie inférieure à 0.0001 % quel que soit le dessin).

Nous pensions en premier lieu que ce problème venait de la taille des datasets (120 000 images pour un dataset quickdraw, 1000 à 2000 avec peu d’images variés pour nos datasets) mais même en répartissant bien les images, le problème persistait.

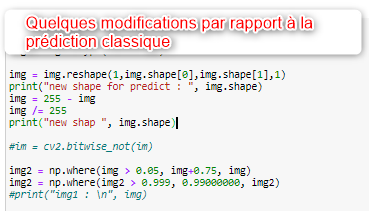
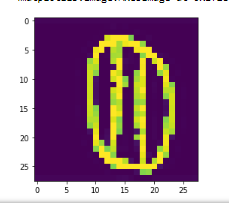
Nous avons donc testés différentes conversions des images avant de les sauvegarder dans des fichiers .npy mais sans succès malheureusement.

Nous avons même remarqué que les images que nous donnions au modèle pour les prédictions étaient différentes des images des datasets originaux.

En effet, il semblait y avoir comme des « traits » supplémentaires (ici image réalisée à la main qui a obtenu une très bonne prédiction par le modèle VS une image du dataset originel) :

 VS 

Avec quelques modifications, notre image ressemblait un peu plus aux images du dataset de quickdraw, cependant les prédictions avec cette dernière était moins bonne (80% comparé au 97% de l’image précédente) :

 🡺 



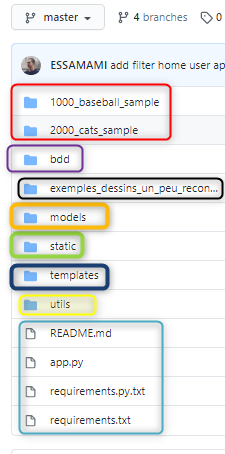
Nous avons essayé d’autres méthodes similaires mais impossible d’avoir une bonne prédiction pour nos datasets composés des dessins des utilisateurs de notre application.

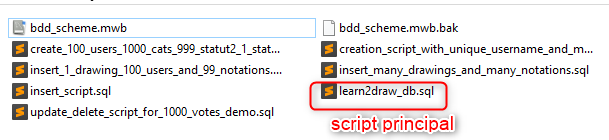
Ce problème a aussi été rencontré par d’autres utilisateurs sur le web, d’autres utilisateurs des datasets quickdraw .npy semblent essayer mais aucune solution évidente n’a été trouvée.

Il pourrait être judicieux de contacter les utilisateurs de forums qui ont rencontré ce problème pour voir s’ils ont trouvé une solution depuis.

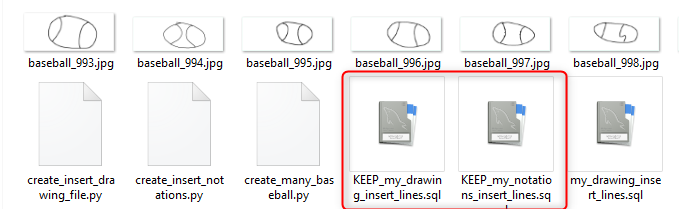
**Guide pour parcourir notre projet (architecture du repo git, liens entre les fonctions, …)**

### **Architecture du repo git**



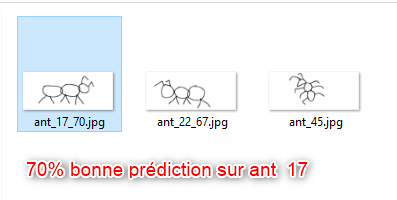


Scripts sql pour créer la base de données et insérer des exemples

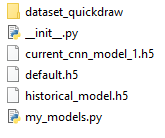
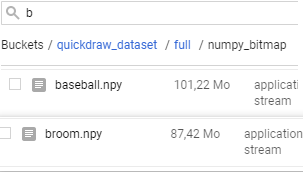
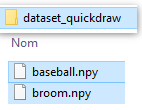


Images à copier dans static/doodle/(category) et scripts pour insérer ces images en base de données (utiles pour créer des datasets .npy)

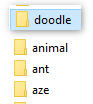
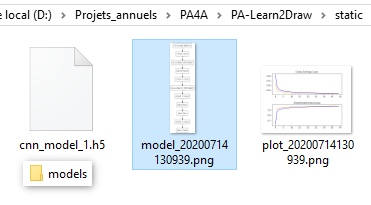
Quelques dessins bien prédits sur certaines catégories (donne des idées de ce qui fonctionne bien)



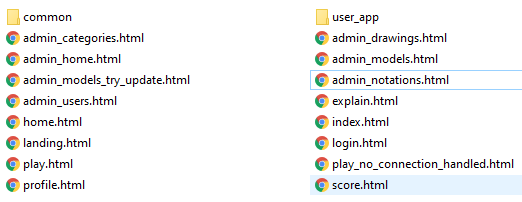
Contient le modèle actuellement utilisé pour prédire (.h5) ainsi que les datasets au format .npy (ils ne sont pas présents sur git, demander le lien ou télécharger au moins broom et baseball sur [ce site](https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap?pli=1))

->  -> 

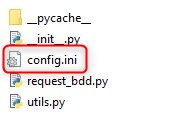
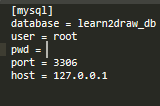
Contient l’ensemble des fichiers CSS, des dessins réalisés par les utilisateurs ainsi que l’ensemble des informations sur les modèles créés

->  -> 

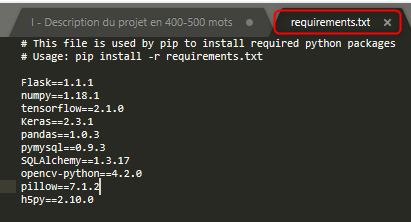
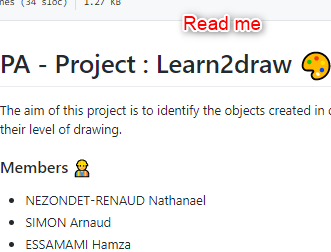
Templates HTML et jinja (ensemble de nos pages web)

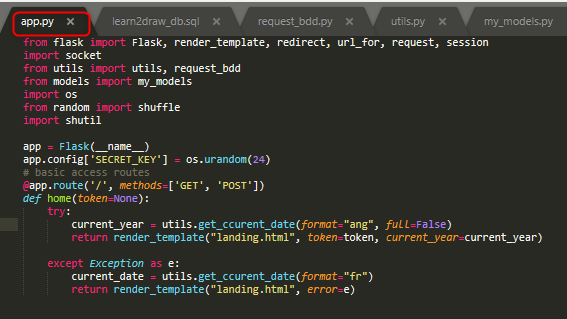


Fichiers python avec quelques configurations utiles ainsi que la plupart de nos requêtes (modifier le fichier config.ini pour mettre les informations de connexion à la base de données)

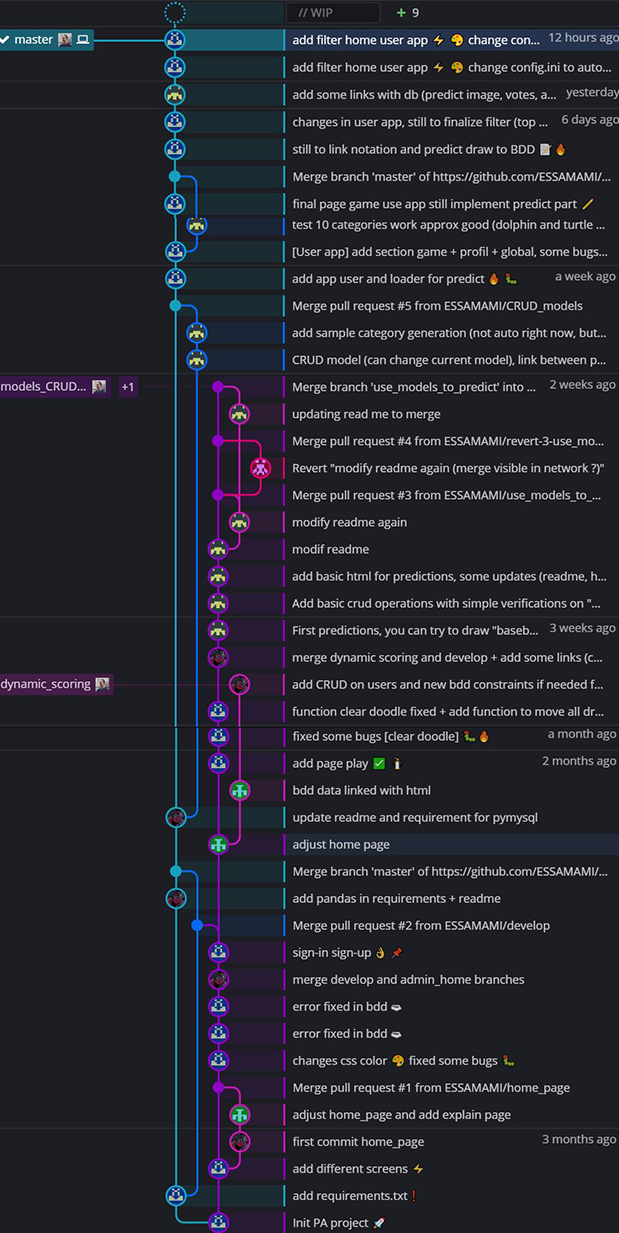
🡪 

Fichiers README avec les instructions à suivre pour démarrer l’application, requirements.txt qui liste l’ensemble des bibliothèques nécessaire ainsi que app.py qui rassemble l’ensemble des routes



### **Arbre des commits**



**Comment bien démarrer l’application ?**

* Installer toutes les bibliothèques présentes dans le fichier requirements.txt
* Récupérer au moins les datasets broom.npy et baseball.npy (sur ce [site](https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap?pli=1))
* Créer une base de données mysql, un schéma learn2draw\_db et exécutez le script **learn2draw\_db.sql**
* Modifiez le fichier utils/config.ini et mettez vos informations de connexion à votre base de données
* Démarrez l’application avec la commande **python app.py**

**Réaliser des tests pour la création d’un nouveau dataset / la validation d’une image (fichiers SQL)**

+ **Test création d’un nouveau dataset.npy**

Pour tester la création d’un nouveau dataset facilement, les données vont être directement insérées en base dans le script de création (learn2draw\_db.sql), il suffira de voter pour le dessin chat 1050 afin d’enclencher la création du dataset numpy et la distribution des points pour la fin de « notation » de l’image (100 votes))

Des vidéos sont également présentes dans le repo git, n’hésitez pas à les visionner.

**Liens principaux entre les fichiers pour nos fonctionnalités principales (au travers d’un exemple)**

**Affichage des données dans la page admin\_user (similaire aux autres)** **puis ajout d’un nouvel utilisateur** :

**1)** route + récupération des informations de notre HTML

**2)** appel de request\_bdd pour obtenir les informations (list\_all\_users)

**3 + 4)** retour des informations dans notre HTML (informations obtenables avec jinja 🡺 {{ variable }})

**5)** Remplir formulaire pour ajout d’utilisateur + Envoi sur une route précise

**6)** Lien avec request\_bdd … 🡺 **Même cycle pour les autres fonctions**

**En images** :

 🡺

2) request\_bdd.py

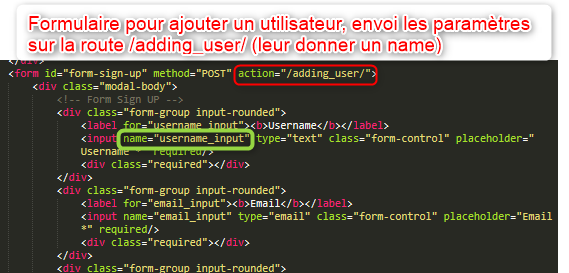
1) app.py

🡺 

3) app.py : Retour au HTML avec les informations



4) template admin\_users : Récupération des informations



5) template admin\_users : Envoi d’information avec formulaire



6) app.py : Récupération des informations et poursuite du cycle (request\_bdd, …)

# VI - Conclusion

**Pistes d’amélioration**

Notre application peut être améliorée sur quelques aspects :

* Les fonctionnalités qui améliorent l’expérience utilisateur
  + En rajoutant une table notifications, les utilisateurs pourront consulter une sorte d’historique et voir par quels moyens ils ont obtenu leurs points
  + Rajouter une galerie dans laquelle un utilisateur pourra sauvegarder ses 30 – 40 dessins favoris
  + Créer un tchat général pour que les utilisateurs puissent discuter entre eux
  + …
* Améliorer les prédictions
  + Essayer de régler le problème des datasets constitués des dessins de nos utilisateurs
  + Créer de meilleurs modèles pour obtenir de meilleures prédictions
* Automatiser certaines tâches en arrière-plan (en utilisant [Celery](https://blog.miguelgrinberg.com/post/using-celery-with-flask) par exemple)
  + Entrainer le modèle actuellement utilisé régulièrement (toutes les 12H ?)
  + Déclencher la création d’un nouveau dataset en tâche de fond pour ne pas gêner l’utilisateur
  + …

**Brève conclusion**

Ce projet a été assez complexe sur certains points mais il a été très formateur pour notre équipe, nous avons pu créer une application intéressante qui même si elle a quelques défauts, est assez facilement améliorable et nous servira de tremplins pour nos futurs projets.

Nous prévoyons de l’améliorer encore à l’avenir afin d’avoir une application de référence à présenter à nos futurs employeurs.

Merci d’avoir pris de votre temps pour vous intéresser à notre projet et pour consulter ce document.

N’hésitez pas à nous contacter à l’une des adresses email suivantes si vous avez des questions :

* [arnaudsimon091@gmail.com](mailto:arnaudsimon091@gmail.com)
* [nat.nez.ren@gmail.com](mailto:nat.nez.ren@gmail.com)
* [hamza.essamami.sio@gmail.com](mailto:hamza.essamami.sio@gmail.com)