# 1 Introduction au problème

Le but de ce travail est de pouvoir visualiser et gérer l'affichage de divers flux d'information sur les écrans des différents campus de la HEIG, comme par exemple les horaires de cours, les news de la RTS, les réservations de salles, etc, à travers une interface web.

# 2 Contraintes et besoins

Les besoins principaux de cette application sont les suivants :
- Gestion de l'affichage de flux d'information sur des écrans dans la HEIG-VD (smartTV ou ordinateur) par une interface web.
- Protocole concis de communication entre les écrans et le backend limitant les échanges.
- Affichage de flux "controlés" (générés par l'application, par exemple un flux RSS) et "non-controlés" (flux de la RTS, horaires des cours, etc).
- Un scheduler qui s'occupe de changer les flux affichés selon un horaire prédéfini.
- Une modélisation générique des flux et une factorisation de ceux venant de sources externes afin de pouvoir les envoyer de la même manière aux écrans
- La possibilité de diffuser plusieurs types de médias (images, vidéos, etc)
- La possibilité de passer outre le scheduler et d'afficher un flux voulu (annonce importante, etc) avec reprise de l'exécution prévue par la suite.

**Quelques précisions :** Un scheduler est un "horaire" qui gère le changement de flux selon un ordre choisi par l'utilisateur. Il n'y aura pas qu'un seul scheduler mais plutôt la possibilité d'en créer plusieurs qui soient assignables à un écran ou groupe d'écrans.

Les contraintes principales quand à elles sont les suivantes :
- Le rendering des flux doit être fait dans un navigateur supportant le Javascript, HTLM5 et CSS3.
- Il doit y avoir une base de données qui enregistre les utilisateurs ainsi que les écrans.
- Il y a plusieurs types d'utilisateurs qui, selon leur emplacement (campus) et/ou leur niveau d'autorisation, peuvent modifier l'affichage des écrans.
- Le système doit être tolérant face aux panne, avec une reprise automatique.
- Le système doit disposer d'une interface simple et être utilisable par des gens du domaine et par des personnes non-initiées.

# 3 Fonctionnalités

Les fonctionnalités nécessaires et principales du programme sont divisées en plusieurs catégories :
- **Frontend**
  1. Interface de login et register sur le site (register dans le cadre du TB)
  2. Ecrans
     (a) Visualisation des écrans actifs et de leur emplacement sur le campus (une "carte" par site)
     (b) Visualisation des informations d'un écran spécifique (et groupe d'écrans)
     (c) Modification de la diffusion actuelle sur un écran/groupe d'écrans (flux "hors-schedule", attribution à un scheduler, arrêt de la diffusion, ...)
  3. Flux
     (a) Operations CRUD sur les flux (interface de création)
     (b) Visualisation des flux utilisables par le système et infos sur leur contenu
  4. Schedulers
     (a) Operations CRUD sur les schedulers (interface de création)
     (b) Visualisation des schedulers utilisables par le système et infos sur leur contenu et horaire
- **Backend - Play !**
  1. Ecrans

   (a) Opérations CRUD sur les écrans
   (b) Emission de token d'authentification pour la connexion des écrans
2. Flux
   (a) Opérations CRUD sur les flux
   (b) Diffusion de flux aux écrans selon un scheduler
   (c) Diffusion de flux hors-schedule (annonces, alertes, etc)
   (d) Formatage et mise en page des flux externes (RTS ou autre)
3. Schedulers
   (a) Opérations CRUD sur les schedulers
   (b) Assignation d'un scheduler à un écran/groupe d'écrans
4. Utilisateurs
   (a) Register
   (b) Login
   (c) Niveaux d'autorisation

— **Backend - Python**
1. Login Service : service python d'authentification des flux basé sur leur adresse MAC et de redirection sur le serveur Play !.

— **Scripting**
1. Script Javascript de connexion des écrans au backend (script exécuté par les écrans pour se connecter au backend et recevoir un flux)

— **Base de données**
1. Utilisateurs avec différents niveaux d'autorisations
2. Ecrans, avec leurs caractéristiques et emplacement
3. Flux utilisés par le système
4. Schedulers de flux

Les fonctionnalités suivantes sont considérées comme secondaires et seront réalisées si le temps le permet :

— **Frontend**
1. Modification en live du contenu d'un flux

— **Backend**
1. Monitoring de l'état des écrans (log du dernier échange avec l'écran)
2. Schedul-ception (scheduler dans un scheduler)

# 4 Echéancier

Le travail sera divisé en 4 parties :

— **Analyse et Modélisation** (2 semaines)
Analyse des contraintes et besoins du travail et modélisation d'un système parvenant à y répondre (schémas de DB, modélisation des flux, etc)

— **Architecture** (1 semaine)
Création d'une architecture de code permettant la réalisation d'un programme efficace, modulable et améliorable par la suite

— **Développement et Tests** (7 semaines)
Codage de l'application, en commençant par le backend et le scripting et en finissant avec le frontend. Création de la base de données en parallèle. Tests unitaires et fonctionnels (60-70% de coverage visé)

— **Rapport et Documentation** (3 semaines)

Total : env. 13 semaines à partir du 25.02