CPOA

Rapport d’analyse

Alexis PERROT, Johann PISTORIUS & Jean Rimlinger

G2S3

Table des matières

[Présentation de l’application 2](#_Toc469265385)

[Diagramme de Classe 3](#_Toc469265386)

[Diagramme de Cas d’Utilisation 4](#_Toc469265387)

[Quelques scénarios de cas d’utilisation 5](#_Toc469265388)

[Gérer les VIP 5](#_Toc469265389)

[Ajouter une fiche VIP 5](#_Toc469265390)

[Modifier une fiche VIP 5](#_Toc469265391)

[Supprimer une fiche VIP 5](#_Toc469265392)

[Ajouter une projection pour un film 5](#_Toc469265393)

[Générer un planning automatiquement 5](#_Toc469265394)

[Ajouter une projection 5](#_Toc469265395)

[Supprimer une projection 5](#_Toc469265396)

[Diagrammes de séquence 6](#_Toc469265397)

[Maquettes 7](#_Toc469265398)

[Page web de gestion des VIP 7](#_Toc469265399)

[Appli de gestion du planning des projections 7](#_Toc469265400)

[Script de création de tables relationnelles 8](#_Toc469265401)

[Squelette de 3 classes JAVA 9](#_Toc469265402)

[Classe Film 9](#_Toc469265403)

[Classe Personne 9](#_Toc469265404)

[Classe VIP 9](#_Toc469265405)

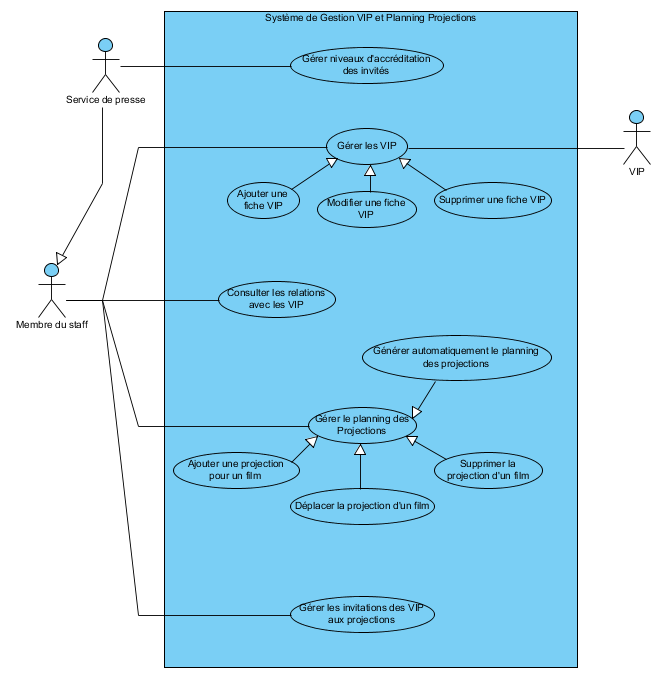
# Présentation de l’application

Le système que nous allons créer pour l’organisation du festival sera divisé en deux parties :

* D’abord un site web qui permettra de créer, modifier et supprimer des fiches VIP rassemblant toutes les informations nécessaires aux festivals concernant les invités les plus important
* Puis un logiciel codé en JAVA qui permettra la planification des projections de films tout au long du festival en respectant bien toutes les contraintes imposées (nombre de films par jour, jury, compétitions), c’est-à-dire la génération automatique d’un planning et l’ajout ou la suppression de projection

# Diagramme de Classe

# Diagramme de Cas d’Utilisation



# Quelques scénarios de cas d’utilisation

## Gérer les VIP

Sur le site web, le membre du staff pourra visualiser la liste des différents VIP déjà enregistrés sur la base de données dans un tableau. Sur le côté il aura accès à trois encadrés lui permettant respectivement de :

Ajouter une fiche VIP : en remplissant des champs puis en cliquant sur le bouton « Ajouter »

Modifier une fiche VIP : en fournissant l’identifiant du VIP à modifier (visible dans le tableau) ainsi que les nouvelles informations le concernant, puis en cliquant sur le bouton « modifier », avec demande de confirmation

Supprimer une fiche VIP en sélectionnant le VIP correspondant puis en cliquant sur le bouton « supprimer », avec demande de confirmation

## Ajouter une projection pour un film

Sur l’application, le membre du staff pourra visualiser le planning dans une liste des jours, avec pour chaque jour une liste des projections (titre du film et horaires). Il aura alors le choix entre :

Générer un planning automatiquement : en cliquant sur le bouton « Générer automatiquement » avec demande de confirmation (car le planning généré écrasera l’ancien s’il y en avait un)

Ajouter une projection : en sélectionnant un film (titre + durée) dans une liste et saisir la date et l’heure du début de la projection, puis cliquer sur « Ajouter »

Supprimer une projection : en sélectionnant la projection dans la liste et en cliquant sur le bouton « Supprimer », avec demande de confirmation

# Diagrammes de séquence

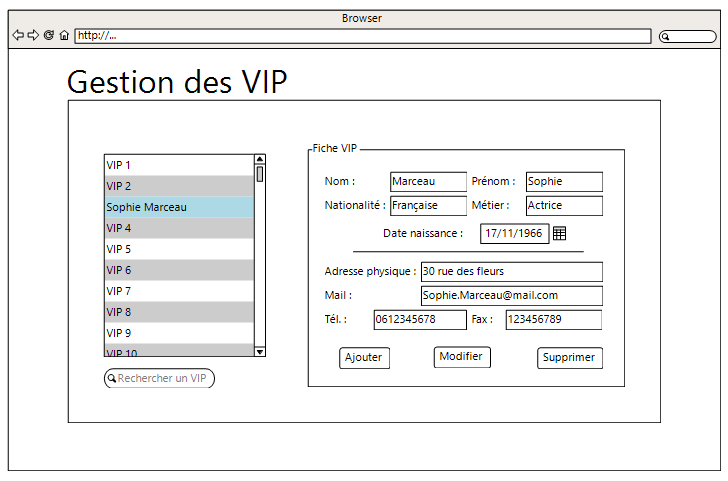
## Gérer les VIPGérer le planning

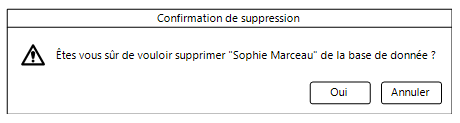
## 

## 

# Maquettes

## Page web de gestion des VIP





## Appli de gestion du planning des projections

# Script de création de tables relationnelles

|  |  |
| --- | --- |
| CREATE TABLE Competition(  numCompetition NUMBER(1),  label VARCHAR2(30)  ); | CREATE TABLE Projection(  numProjection NUMBER(4),  dateDebut DATE,  numFilm NUMBER(4),  numSalle NUMBER(4),  CONSTRAINT fk\_numCompet FOREIGN KEY (numFilm) REFERENCES Film(numFilm),  CONSTRAINT fk\_numSalle FOREIGN KEY (numSalle) REFERENCES Salle(numSalle)  ); |
| CREATE TABLE Film(  numFilm NUMBER(4),  duree NUMBER(5),--En minutes  realisateur VARCHAR2(30),  numCompetition NUMBER(1),  CONSTRAINT fk\_numCompet FOREIGN KEY (numCompetition) REFERENCES Competition(numCompetition)  ); | CREATE TABLE Salle(  numSalle NUMBER(4),  nom VARCHAR2(30),  places NUMBER(4)  ); |
| CREATE TABLE Personne(  numPersonne NUMBER(4),  nom VARCHAR2(30),  prenom VARCHAR2(30),  numTel NUMBER(10),  adressePhysique VARCHAR2(300),  mail VARCHAR2(100),  fax NUMBER(10)  ); | CREATE TABLE VIP(  numVIP NUMBER(4),  nationalite VARCHAR2(30),  metier VARCHAR2(30),  dateNaissance date,  numPersonne NUMBER(4),  CONSTRAINT fk\_numPersonne FOREIGN KEY (numPersonne) REFERENCES Personne(numPersonne)  ) ; |
| CREATE TABLE Presse(  niveauAccreditation NUMBER(1),  numPersonne NUMBER(4),  CONSTRAINT fk\_numPersonne FOREIGN KEY (numPersonne) REFERENCES Personne(numPersonne)  ); | CREATE TABLE Jure(  numVIP NUMBER(4),  numCompetition NUMBER(1),  CONSTRAINT fk\_numCompetition FOREIGN KEY (numCompetition) REFERENCES Competition(numCompetition),  CONSTRAINT fk\_numVIP FOREIGN KEY (numVIP) REFERENCES VIP(numVIP)  ); |
| CREATE TABLE Celebrite(  numVIP NUMBER(4),  photo BINARY,  coefImportance NUMBER(1),  compagnon VARCHAR2(30),  CONSTRAINT fk\_numVIP FOREIGN KEY (numVIP) REFERENCES VIP(numVIP)  ); | CREATE TABLE Echange(  numEchange NUMBER(10),  numVIP NUMBER(4),  modeCommunication VARCHAR2(30),  contenu BINARY,  CONSTRAINT fk\_numVIP FOREIGN KEY (numVIP) REFERENCES VIP(numVIP)  ); |

# Squelette de 3 classes JAVA

## Classe Film

package festival.de.cannes;

public class Film {

private int duree;

private int numFilm;

private String realisateur;

public void getFilm(){

}

}

## Classe Personne

package festival.de.cannes;

public class Personne {

private int numPersonne;

private int numTel;

private String adressePhysique;

private String mail;

private int fax;

public void getPersonne(){

}

}

## Classe VIP

package festival.de.cannes;

public class VIP extends Personne{

private String nationalite;

private String metier;

private String dateNaissance;

@Override

public void getPersonne(){

}

}