TELECOM SudParis

PROJET INFORMATIQUE 1ERE ANNEE

PRO3600

VOTE ELECTRONIQUE A LA CONDORCET RANDOMISE

Estelle CANOVAS
Julien DENIZE
Rémi MARSAL
Théo PEROCHON

Enseignant Responsable : Paul GIBSON

Table des matières

Introduction

Ce document décrit le développement d'une interface web de vote électronique répondant au scrutin de Condorcet randomisé.

Avec cette méthode de scrutin, les votants sont doivent classer leurs choix selon leur ordre de préférence. Une fois que le vote est clos, un algorithme confronte sous forme de duels les choix faits par les utilisateurs, afin de déterminer le vainqueur de Condorcet, c'est-à-dire l'alternative qui comparé tour à tour à toutes les autres, s'avère être à chaque fois la préférée. Il est possible que plusieurs vainqueurs existent – c'est le paradoxe de Condorcet. Le vainqueur final est alors désigné selon une loi de probabilité parmi les candidats en tête des votes.

Pour notre plateforme web, nous avons décidé de proposer deux types de votes : le vote privé et le vote public.
Pour le vote privé, l'administrateur du vote peut définir au préalable l'ensemble des personnes autorisées à voter. Les votants doivent alors avoir un compte sur le site pour pouvoir accéder aux votes auxquels ils sont habilité à voter.
Pour le vote public, avoir un compte n'est pas nécessaire, n'importe qui peut y participer.

Pour voter, l'utilisateur classe les différentes alternatives qu'il juge conforme à ces idées, en accordant à chacune d'entre elles un rang, dans l'ordre de ses préférences.

Du point de vue administrateur le déroulement d'un vote consiste à créer le vote en lui attribuant un objet, en proposant une description facultative, en définissant le type de vote (public ou privé), la date exacte de clôture du vote, les différentes alternatives, et les utilisateurs autorisés à voter s'il s'agit d'un vote privé. Il obtient le résultat final du vote à l'issue de la date de clôture.

Du point de vue utilisateur, le déroulement du vote consiste à y accéder soit par un lien soit en parcourant les votes qui lui sont accessibles (publics ou privés) puis de voter selon ses préférences.

Enfin, les votes et les accès doivent être sécurisés.

A la suite de cette introduction, le document comporte les parties suivantes :

- Une présentation du cahier des charges ;
- La description du développement de l'interface ;
- Un manuel utilisateur.

Une conclusion et une annexe fermeront ce document.

Le développement de cette interface a été réalisé dans le cadre du projet informatique de première année de Télécom SudParis par Estelle Canovas, Julien Denize, Rémi Marsal, Théo Perochon, sous l'encadrement de Paul Gibson, professeur du département informatique de Télécom SudParis.

Cahier des charges

Cette partie se décompose en quatre sous-parties :

- Analyse des besoins ;
- Fonctionnalités avec contraintes et responsables ;
- Architecture de la plateforme web ;
- Langages utilisés.

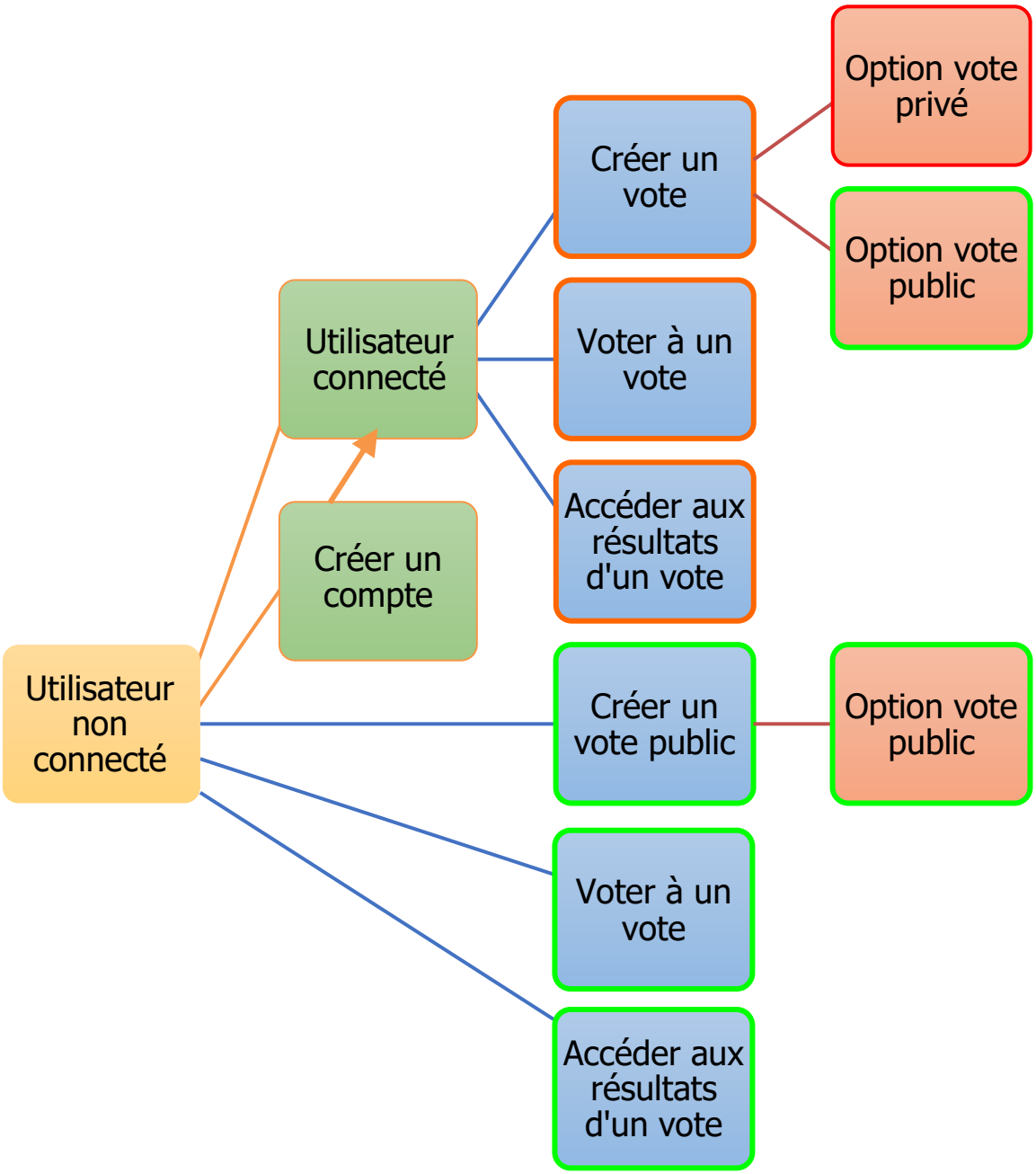
Analyse des besoins

Dans notre démocratie actuelle, le vote occupe une place centrale puisqu'il permet d'élire nos représentants. Cependant les suffrages en place ne permettent pas de satisfaire à chaque fois le principe de Condorcet. Or ce principe désigne comme vainqueur d'un scrutin le candidat qui, comparé à n'importe quel autre, s'avère à chaque fois être le candidat préféré des votants. C'est un principe fondamental que doit vérifier chaque vote pour être démocratique. Dans ce contexte, nous proposons de réaliser une plateforme web permettant d'organiser un vote qui satisfait le principe de Condorcet dans tous les cas.

Fonctionnalités

	Contraintes
Algorithme de vote	Condorcet randomisé
Plateforme	Site web
Vote public	-Créer un vote par un administrateur -Générer un lien partageable pour accéder au vote -Le votant vote
Vote privé	-Créer un vote par un administrateur ayant un compte -L'administrateur liste les personnes autorisées à voter -Générer un lien partageable pour accéder au vote -Notifier les votants de la création d'un vote sur leur compte et par mail -Le votant peut accéder au vote via son compte si autorisé -Le votant vote
Sécurisation	-Empêcher la récupération des données privées des utilisateurs et des votes -Empêcher un utilisateur de voter plusieurs fois en rafraichissant la page -Protéger la plateforme web de différentes attaques comme l'injection SQL -Blockchain (si le temps le permet)

Architecture de la plateforme web



Voici le détail des différentes pages :

Nom de la page	Contenu de l'interface
Utilisateur non connecté	-barre de recherche pour trouver des votes publics -barre de connexion -bouton créer un nouveau compte -liste des votes répartis sur plusieurs pages accessibles en cliquant sur la numérotation des pages -onglet créer un vote

Utilisateur connecté	-barre de recherche pour trouver des votes publics -onglet contenant la liste des votes répartis sur plusieurs pages accessibles en cliquant sur la numérotation des pages -onglet contenant la liste des votes auxquels il a accès en spécifiant s'il n'a pas encore voté, s'il a voté mais que le vote n'est pas clôturé et si le vote est clôturé -onglet de déconnexion -bouton valider
Créer un compte	Zone de texte : -pseudo -adresse mail -mot de passe -confirmation du mot de passe -bouton valider
Créer un vote	-objet du vote -les différentes alternatives -cocher l'option vote privé ou vote public -choisir la date d'ouverture du vote -choisir la date de clôture du vote -bouton valider
Créer un vote public	-objet du vote -les différentes alternatives -choisir la date d'ouverture du vote -choisir la date de clôture du vote -bouton valider
Option vote privé	-formulaire à remplir avec les adresses des personnes autorisées à voter -affichage du lien du vote -bouton valider
Option vote public	-affichage du lien du vote -bouton valider
Voter	-affichage de l'objet du vote -pour chaque position dans le vote, le votant sélectionne l'alternative qu'il souhaite dans un menu déroulant -bouton valider
Accéder aux résultats d'un vote	-afficher l'objet du vote -afficher les résultats du vote sous forme d'un camembert

Langages utilisés

L'algorithme de vote sera écrit en Java.

La plateforme web sera écrite en PHP, HTML, CSS, SQL.

Développement

Cette partie s'organise de la manière suivante :

- Analyse du problème et spécification fonctionnelle ;
- Conception préliminaire puis détaillée ;
- Codage ;
- Tests.

Analyse du problème et spécification fonctionnelle

Après une réflexion plus poussée sur le cahier des charges, au sein du groupe et avec notre professeur encadrant, nous avons pu définir avec précision les différentes fonctionnalités attendues de notre plateforme et les objets intervenant.

Comment voter ? Est-il possible de voter à main levée ou par reconnaissance vocale ?

Les votes se feront uniquement via l'interface web. Chaque participant devra voter séparément.

De quoi a besoin l'administrateur pour créer un vote ?

L'administrateur définit un vote avec :

- Le titre du vote ;
- Une description du vote ;
- La date de début du vote ;
- La date de clôture du vote ;
- Le type de vote ;
- Le nombre d'alternatives du vote ;
- Les alternatives.

Est-il possible pour l'administrateur de suivre l'état d'avancement des votes qu'il a créés ?

L'administrateur peut connaître le résultat des votes qu'il a créés seulement à la clôture de ceux-ci.

Quelle est la différence entre un vote public et un vote privé ?

Un vote public est visible et accessible par tous. Chacun peut participer au vote.

Un vote privé est visible uniquement par les membres autorisés par l'administrateur à la création du vote.

Quelles sont les pages accessibles sur la plateforme web ?

Les pages doivent permettre à l'utilisateur de créer un compte, s'inscrire, créer un vote, voir tous les votes, voter, voir le résultat des votes auxquels il a accès.

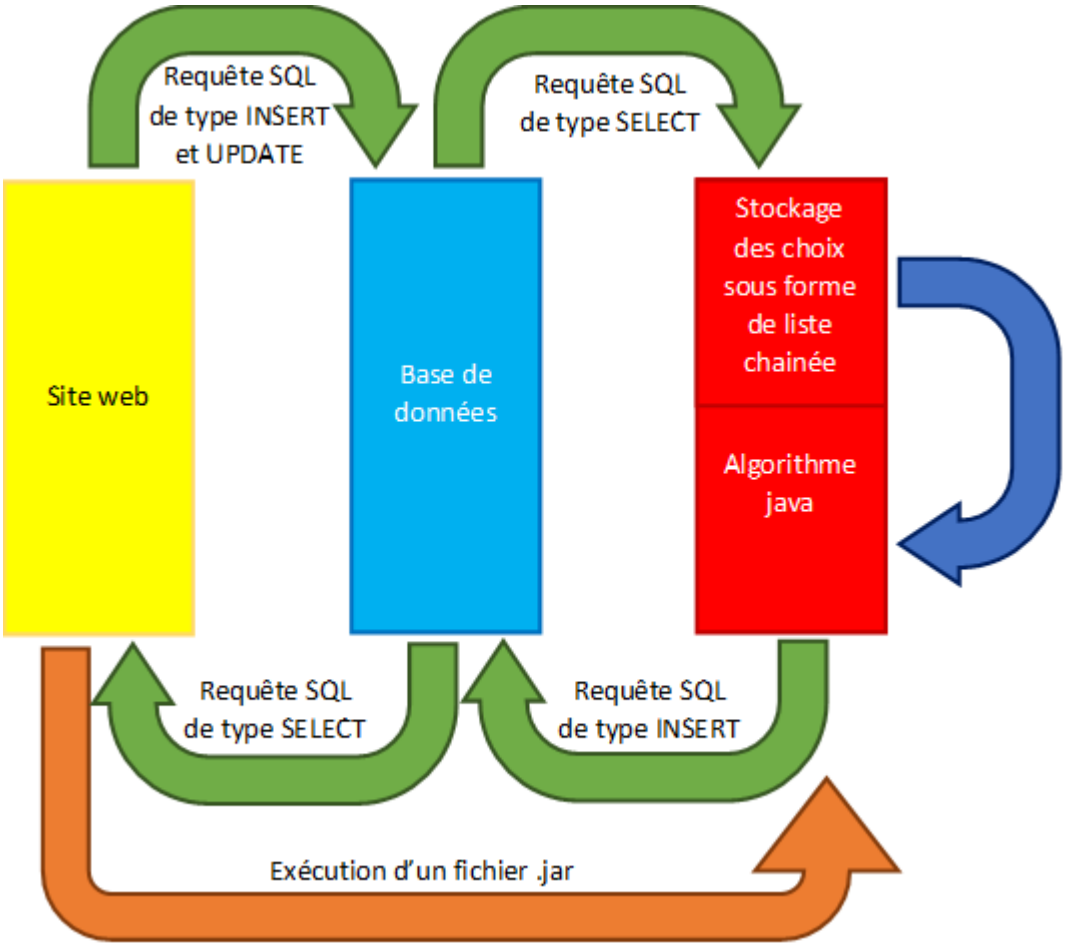
Comment vont communiquer les différentes parties du code ?

La plateforme web va communiquer avec la base de données, pour tout ce qui concerne les votes et les utilisateurs. Elle va aussi communiquer avec l'algorithme de vote qui va déterminer le vainqueur d'un vote en interaction avec la base de données.

Nous pouvons alors préciser les fonctionnalités et leurs conditions d'utilisation :

- S'inscrire :
    - Le pseudo ne doit pas avoir été déjà créé ;
    - L'adresse email ne doit pas avoir déjà servi ;
    - Les champs Pseudo, Email, Mot de passe doivent être remplis ;
- Se connecter :
    - Les champs Email et Mot de passe doivent être corrects ;
- Se déconnecter : L'utilisateur doit être connecté ;
- Créer un vote :
    - L'utilisateur doit être connecté ;
    - Les champs suivants doivent être non vides :
        - Titre du vote,
        - Description du vote,
        - Date de début,
        - Date de fin,
        - Type du vote,
        - Nombre d'alternatives,
        - Alternatives ;
- Consulter les votes :
    - Tous les votes publics doivent être affichés sur la page ;
    - L'utilisateur doit être connecté pour visualiser les votes privés auxquels il a droit ;
- Voter :
    - L'utilisateur doit être connecté dans le cas d'un vote privé ;
- Consulter les résultats d'un vote :
    - L'utilisateur doit pouvoir consulter les résultats des votes auxquels il a droit ;
- Clôturer un vote :
    - L'administrateur d'un vote doit pouvoir clôturer un vote qu'il a créé ;
    - Le vote doit se clôturer automatiquement à la date de fin.

Conception préliminaire



Architecture application web

Pour développer notre plateforme de vote en ligne, nous avons le site web qui permet au client de faire des requêtes, créer un vote, voter et afficher les résultats. Ce site communique avec une base de données et un algorithme qui permettent d'administrer les votes, les élections, les utilisateurs et de créer une application web répondant au cahier des charges en intégrant différents modules spécifiques.

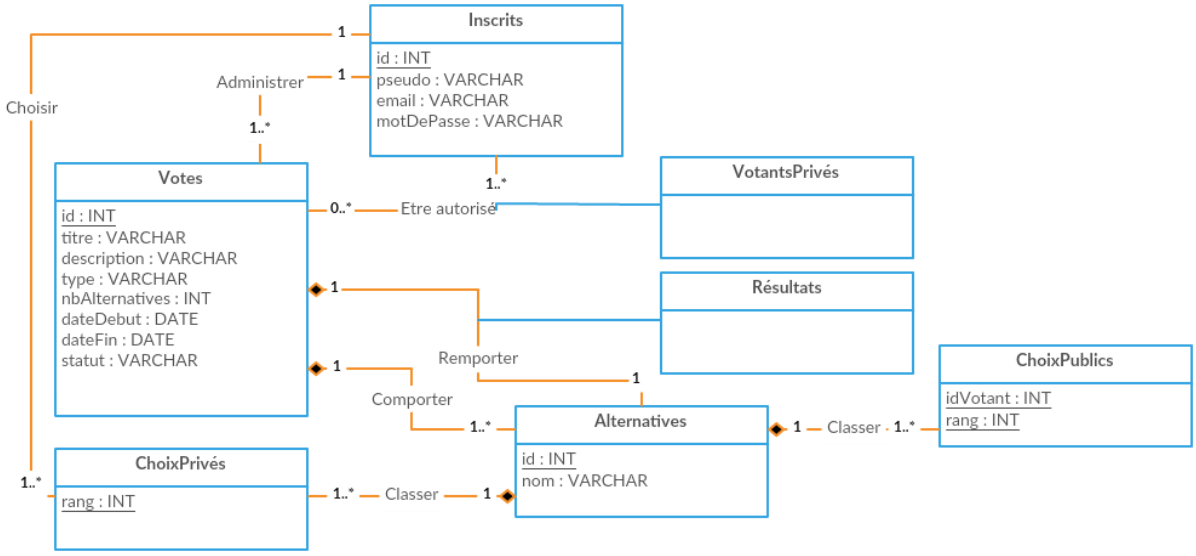


Diagramme de classes de la base de données

Pour pouvoir administrer les votes et les utilisateurs, il nous faut une base de données permettant d'accéder aux données et les faire communiquer facilement. Le diagramme ci-dessus modélise les interactions entre les différentes relations qui composent notre base de données.

On peut noter par exemple qu'un vote a besoin d'un administrateur. De plus, pour accéder à un vote il faut que l'utilisateur y ait droit. Enfin, les choix publics et privés sont séparés.

Présentation de chaque classe SQL

Inscrits : représente un utilisateur s'étant inscrit avec son pseudo, email et mot de passe.

Votes : représente un suffrage avec les attributs définis par le cahier des charges.

Alternatives : représente une alternative d'un vote.

ChoixPublics : représente le classement d'une alternative de la part d'un utilisateur non connecté dans un vote.

ChoixPrivés : représente le classement d'une alternative de la part d'un utilisateur connecté dans un vote.

VotantsPrivés : représente un votant autorisé dans un vote privé.

Résultats : représente le résultat d'un vote.

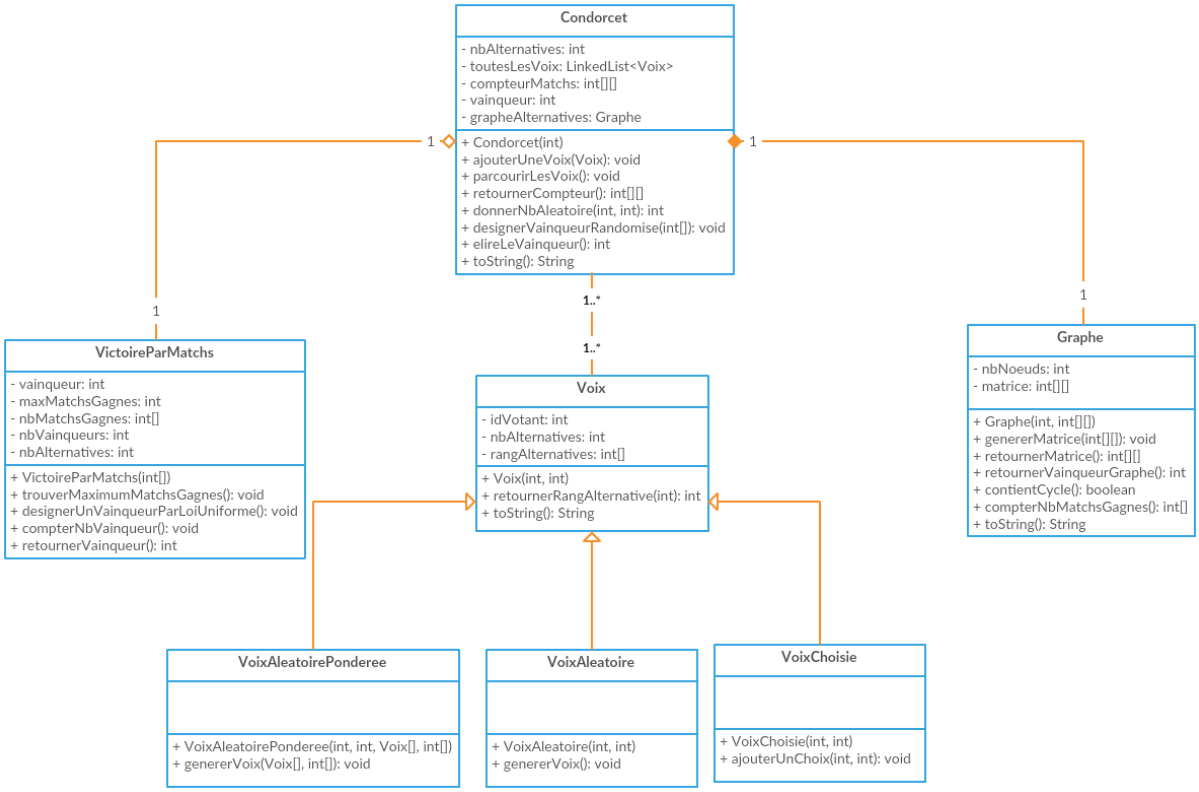


Diagramme de classes de l'algorithme de Condorcet

L'algorithme de Condorcet randomisé est complexe et nécessite l'utilisation de différentes classes afin de structurer et clarifier le code, dans le but de faciliter le développement de l'algorithme.

On peut noter que la classe Condorcet représente une élection et est au cœur de l'algorithme, puisqu'elle est le seul lien entre toutes les autres classes.

Présentation de chaque classe algorithme

Condorcet : représente une élection d'un suffrage.

Graphe : représente le graphe orienté des matchs entre les différentes alternatives du vote.

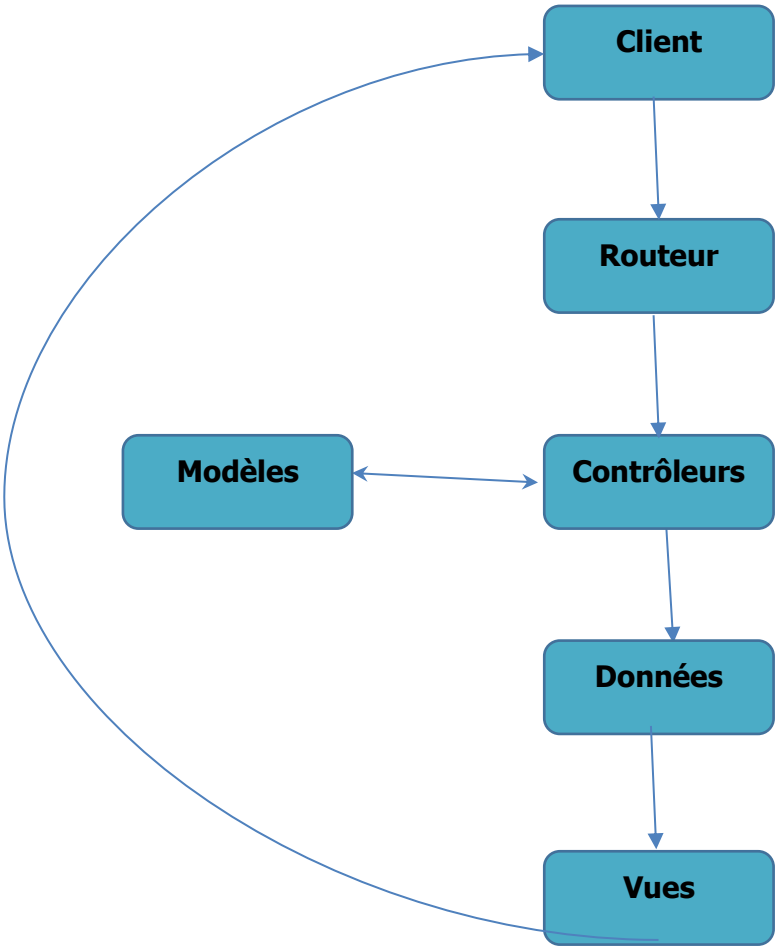
VictoireParMatchs : représente les vainqueurs en tête de l'élection s'il n'y a pas de vainqueur de Condorcet ni de cycle dans le graphe et permet d'en retirer un vainqueur unique.

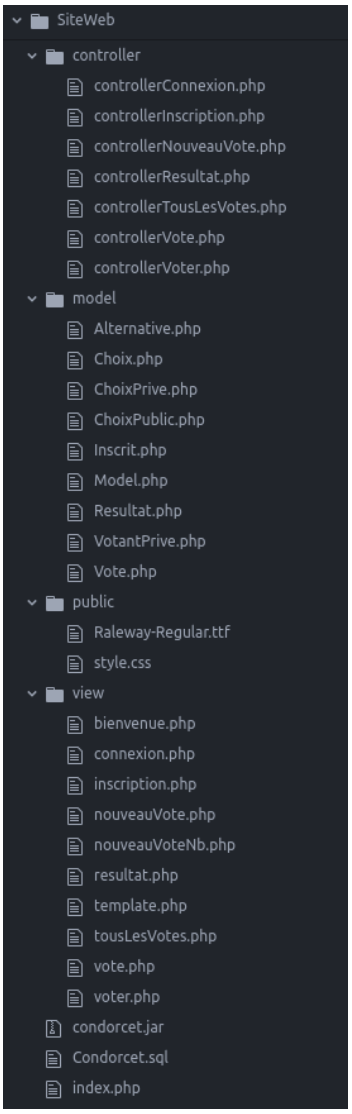
Voix : représente la voix d'un votant, c'est-à-dire tous les choix de classement qu'il a entrés pour les différentes alternatives.

VoixAleatoirePonderee : représente une voix définie aléatoirement par pondération de différentes voix données.

VoixAleatoire : représente une voix définie aléatoirement.

VoixChoisie : représente une voix rentrée dans la base de données.





Architecture du site Web

<u>Conception détaillé</u>

Pour développer Le site web, nous avons utilisé une structure MVC (Modèle-Vue-Contrôleur) comme présenté par le schéma précédent.

On a donc un routeur qui traite la requête HTML du client et appelle le bon contrôleur et la bonne fonction suivant l'action demandée. Ce contrôleur récupère les données dans la base de données si elles sont nécessaires, en demandant au modèle ces données. Le contrôleur envoie ces données à une vue qui est présentée au client, qui obtient donc les informations qu'il avait demandée dans un premier temps au routeur.

Exemple de requête client

Prenons l'exemple d'un client qui souhaite afficher tous les votes auxquels il a accès. La requête s'écrira selon la forme http://localhost/SiteWeb/index.php?action=tousLesVotes.

Le routeur stocke la variable action et regarde si elle correspond à une action qu'il est sensé connaître. Si ce n'est pas le cas, il affiche une erreur et renvoie à la page d'accueil.

```php
elseif($action == 'tousLesVotes') {
  //appelle le controleur pour gérer les votes en cours.
  require('./controller/controllerTousLesVotes.php');
  getTousLesVotes();
}
```

Routeur index.php pour l'action tousLesVotes

On voit ici que l'action tousLesVotes est connue et que le routeur charge le contrôleur intitulé controllerTousLesVotes et appelle sa fonction getTousLesVotes.

```php
<?php

/**
  * Controller pour gérer les votes en cours.
  *
  * Le client demande à voir les différents votes en cours.
  * @author julien
**/

require_once('./model/Vote.php');
require_once('./model/Inscrit.php');

/**
  * Envoie les différents votes en cours au client.
**/
function getTousLesVotes() {
  if(isset($_SESSION['pseudo'])) {
    $votes = Vote::recupererVotes(new Inscrit($_SESSION['pseudo']));
  }
  else $votes = Vote::recupererVotes();
  require('./view/tousLesVotes.php');
}
```

controllerTousLesVotes.php

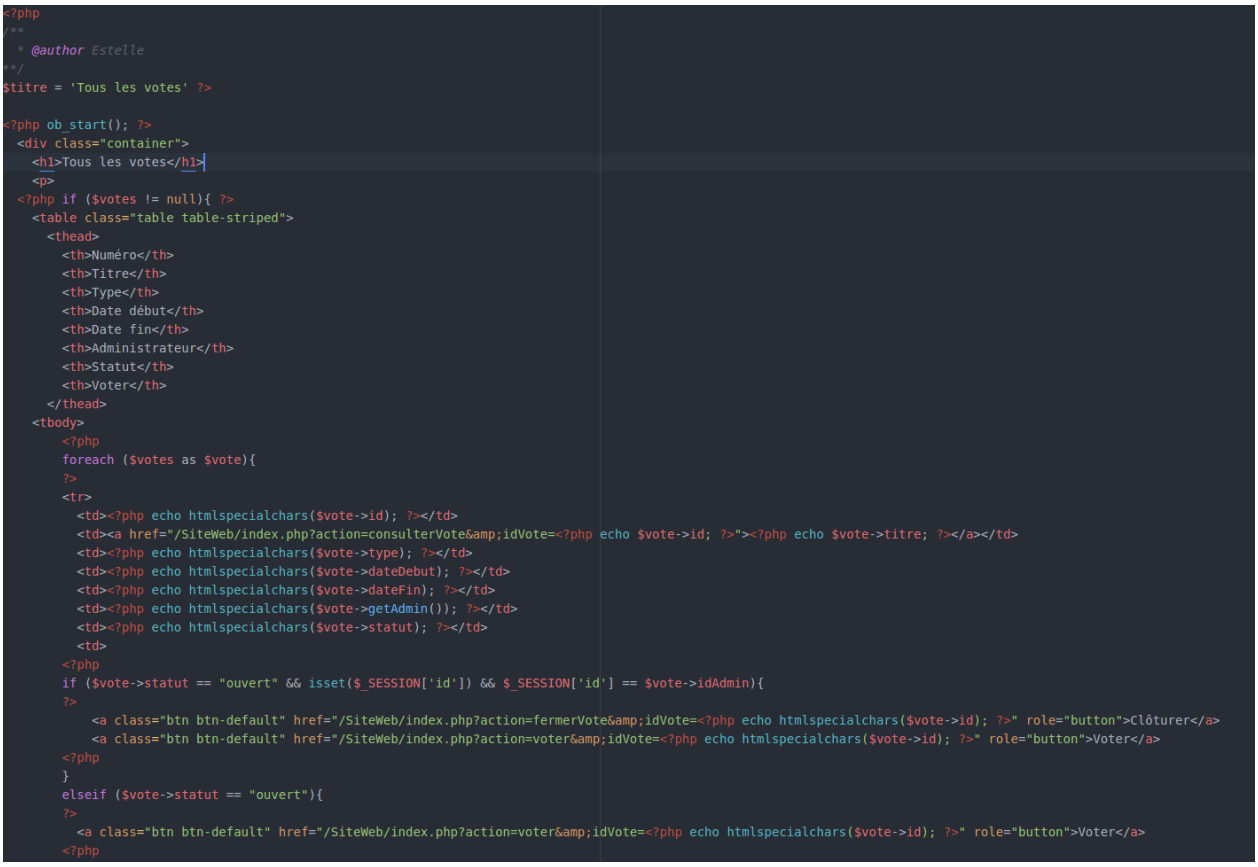
controllerTousLesVotes charge les classes du modèle dont il a besoin, c'est-à-dire Vote et Inscrits.

getTousLesVotes récupère tous les votes grâce à la fonction statique recupererVotes de la classe Vote et lui fournit en paramètre le pseudo de l'utilisateur s'il est connecté. Ensuite elle charge la vue qui peut donc utiliser les données récupérées précédemment.

```php
/**
  * Récupère les votes dans la base de données dont l'utilisateur a accès.
  *
  * @param utilisateur null si il n'y a pas d'utilisateur connecté, sinon utilisateur connecté
  * @return array Tableau contenant les votes.
**/
public static function recupererVotes(Inscrit $utilisateur = null) {
  if($utilisateur == null) {
    //Pour pouvoir accéder à la méthode executerRequete
    $utilisateur = new Inscrit(null, null);
    $sql = 'SELECT id, titre, description, type, nbAlternatives, dateDebut, dateFin, statut, idAdmin
            FROM Votes
            WHERE type=\'public\'';
    $req = $utilisateur->executerRequete($sql);
  }
  else{
    //Le OR type='prive' devra être supprimé lorsque les votants pourront être sélectionnés.
    $sql = 'SELECT V.id as id, titre, description, type, nbAlternatives, dateDebut, dateFin, statut, idAdmin
            FROM Votes V JOIN VotantsPrives ON V.id = idVote JOIN Inscrits I ON idInscrit = I.id
            WHERE pseudo = :pseudo
            UNION
            SELECT id, titre, description, type, nbAlternatives, dateDebut, dateFin, statut, idAdmin
            FROM Votes
            WHERE type=\'public\' or type =\'prive\'';
    $parametres = array('pseudo' => $utilisateur->pseudo);
    $req = $utilisateur->executerRequete($sql, $parametres);
  }
  if($req->rowCount() == 0) return null;
  while($donnees = $req->fetch()) {
    $tableau[] = new Vote($donnees['id'], $donnees['titre'], $donnees['description'], $donnees['type'],
                          $donnees['nbAlternatives'], $donnees['dateDebut'], $donnees['dateFin'],
                          $donnees['statut'],$donnees['idAdmin']);
  }
  return $tableau;
}
```

Fonction statique recupererVotes dans le modèle Vote.php

La fonction statique recupererVotes dans la classe Vote demande à la base de données par requête SQL les votes auxquels l'utilisateur a accès. S'il est déconnecté, ce sont les votes publics. S'il est connecté, ce sont les votes publics accompagnés des votes privés dont l'accès lui est autorisé. Cette fonction renvoie dans un tableau tous les votes ainsi récupérés.



```php
<?php
/**
 * @author Estelle
**/
$titre = 'Tous les votes' ?>

<?php ob_start(); ?>
 <div class="container">
   <h1>Tous les votes</h1>
   <p>
 <?php if ($votes != null){ ?>
   <table class="table table-striped">
     <thead>
       <th>Numéro</th>
       <th>Titre</th>
       <th>Type</th>
       <th>Date début</th>
       <th>Date fin</th>
       <th>Administrateur</th>
       <th>Statut</th>
       <th>Voter</th>
     </thead>
   <tbody>
       <?php
       foreach ($votes as $vote){
       ?>
       <tr>
         <td><?php echo htmlspecialchars($vote->id); ?></td>
         <td><a href="/SiteWeb/index.php?action=consulterVote&amp;idVote=<?php echo $vote->id; ?>"><?php echo $vote->titre; ?></a></td>
         <td><?php echo htmlspecialchars($vote->type); ?></td>
         <td><?php echo htmlspecialchars($vote->dateDebut); ?></td>
         <td><?php echo htmlspecialchars($vote->dateFin); ?></td>
         <td><?php echo htmlspecialchars($vote->getAdmin()); ?></td>
         <td><?php echo htmlspecialchars($vote->statut); ?></td>
         <td>
       <?php
       if ($vote->statut == "ouvert" && isset($_SESSION['id']) && $_SESSION['id'] == $vote->idAdmin){
       ?>
           <a class="btn btn-default" href="/SiteWeb/index.php?action=fermerVote&amp;idVote=<?php echo htmlspecialchars($vote->id); ?>" role="button">Clôturer</a>
           <a class="btn btn-default" href="/SiteWeb/index.php?action=voter&amp;idVote=<?php echo htmlspecialchars($vote->id); ?>" role="button">Voter</a>
       <?php
       }
       elseif ($vote->statut == "ouvert"){
       ?>
         <a class="btn btn-default" href="/SiteWeb/index.php?action=voter&amp;idVote=<?php echo htmlspecialchars($vote->id); ?>" role="button">Voter</a>
       <?php
```

Vue tousLesVotes.php

La vue tousLesVotes affiche sous la forme de tableau les votes auxquels l'utilisateur a le droit grâce aux données fournies par le controllerTousLesVotes. Pour les votes ouverts elle permet de demander à voter ; pour les votes clôturés elle permet de demander à afficher les résultats. L'administrateur peut également demander à clôturer le vote s'il est encore ouvert. Toutes ces demandes du client se réalisent par la pression d'un bouton.

Tout le site web repose sur ce principe du pattern MVC. Il contient trop de fonctions et de fichiers de la structure MVC pour que l'on explique chacune d'entre elles ici et c'est pourquoi nous vous avons montré les différents fichiers composant le site web précédemment.

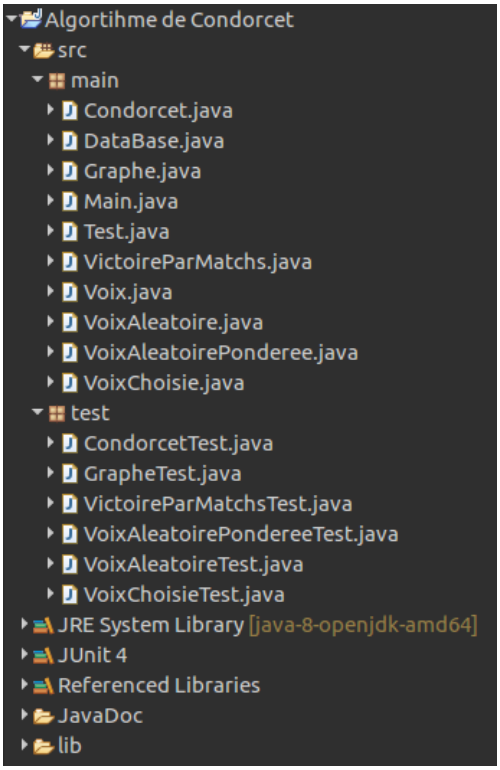
Nous vous invitons donc à parcourir le code qui a été commenté sommairement afin d'aider à la compréhension. De plus les noms de fichiers sont le plus explicite possible et le développement web est suffisamment proche du langage humain pour être facilement compréhensible.

Le rapport ne nous semble pas être le bon support pour expliquer nos fonctions du fait de leur taille. La soutenance sera une meilleure occasion pour faire une réelle démonstration.

Développement de l'algorithme

Pour le développement de l'algorithme, le code a été séparé en deux packages :

- Main qui contient l'algorithme en tant que tel ;
- Test qui contient les classes testant les classes de l'algorithme ;



Structure du code de l'algorithme

Les classes sont celles définies par le diagramme de classes de la conception préliminaire avec les champs et méthodes définis à l'exception de DataBase qui est la classe permettant d'échanger avec la base de données et Main qui à travers sa méthode main, démarre l'algorithme.

```
function fermerVote() {
  $idVote = (int)$_GET['idVote'];
  $vote = new Vote($idVote, null, null, null, null, null, null, null , null);
  if($vote->existeVote() && $vote->statut='ouvert'){
    //$aujourdhui= new DateTime();
    //if ($aujourdhui->diff($vote->dateFin)>0) {
    shell_exec("java -jar condorcet.jar 'jdbc:mysql://localhost:3306/Condorcet' 'root' 'root' $vote->id");
```

controllerVote fonction fermerVote exécute l'algorithme de Condorcet

Lors de la clôture d'un vote, le site web utilise l'algorithme de Condorcet. Il fournit en paramètres les identifiants de connexion à MySQL pour que l'algorithme puisse communiquer avec la base de données et l'id du vote pour lequel l'algorithme va devoir donner le vainqueur.

```java
public static void main(String[] var) throws Exception{
    int idVote = Integer.parseInt(var[3]);
    DataBase db = new DataBase(var[0], var[1], var[2]);
    db.connectDB();
    int n = db.getNbAlternatives(idVote);
    Condorcet election = new Condorcet(n);
    db.remplirToutesLesVoix(idVote, election);
    db.saveWinner(election.elireLeVainqueur(), idVote);
    System.out.println(election.toString());
    db.deconectDB();
```

Méthode main de la classe Main appelé par le site web pour trouver et stocker le vainqueur du vote

Le pseudo code qui suit permet de comprendre comment fonctionne l'algorithme.

idVote = paramètre donné par le site web

nbAlternatives = nbAlternatives stocké dans la table Votes dans la base de données pour idVote

election = new Condorcet(nbAlternatives)

Pour chaque votant du vote récupéré dans la base de données :

Créer une voix contenant tous les choix du votant récupérés dans la base de données, c'est-à-dire le classement de chacune des voix

Ajouter voix dans l'election

Fin pour

Election.elireLeVainqueur()

L'élection compte pour chaque voix et chaque alternative les matchs gagnés d'une alternative face à une autre

L'election construit un graphe des matchs entre les différentes alternatives qui est orienté par des arcs définis par : si l'alternative i bat une alternative j, i pointe j

Si il y a un vainqueur du graphe (une alternative qui bat toutes les autres):

L'élection le désigne comme vainqueur

Sinon si le graphe contient un cycle :

L'élection désigne un vainqueur par loi de probabilité définie par $P(alternative\ i\ gagne) = \frac{nombre\ de\ matchs\ gagnés\ par\ alternative\ i}{nombre\ de\ matchs\ totaux}$

Sinon :

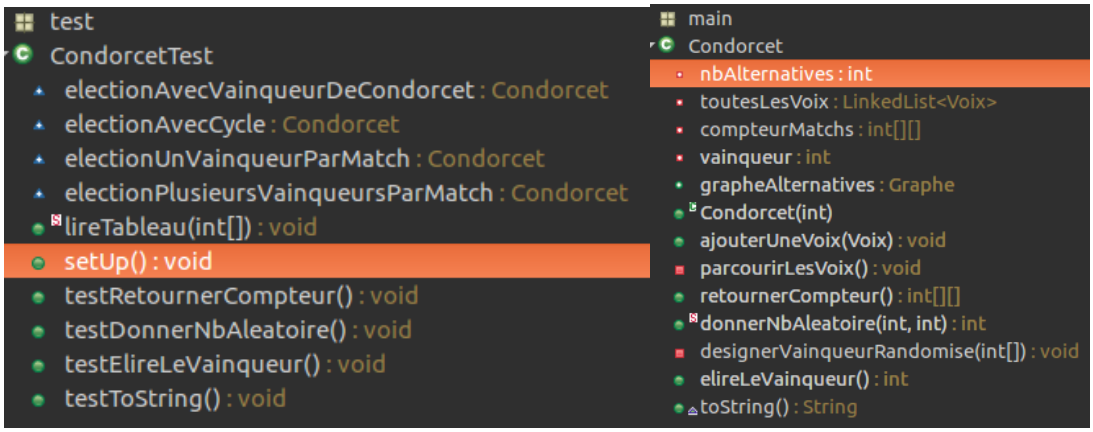
L'élection sélectionne les vainqueurs en tête c'est-à-dire les alternatives ayant gagné le plus de matchs

Elle désigne par loi de probabilité uniforme sur ce nombre d'alternatives le vainqueur

L'election renvoie le vainqueur

Le vainqueur est stocké dans la base de données dans la table Résultats

A titre d'exemple d'une classe de l'algorithme et d'une classe de test, ci-dessous sont montrées la structure de Condorcet et CondorcetTest.



test.CondorcetTest.java et main.Condorcet.java

De même que pour le site web nous vous invitons à parcourir le code pour le comprendre. Le code a été intégralement commenté et la javadoc est également fourni ou générable si vous le désirez.

Codage

Les classes java sont documentées de telle sorte qu'il est possible de générer la javadoc et pouvoir naviguer facilement dans le code de l'algorithme afin de mieux comprendre sa structure et son fonctionnement.

Le site web est également commenté mais dans une moindre mesure puisqu'il existe certains outils pour générer de la phpdoc mais que nous n'avons pas voulu utiliser car ils requièrent une certaine écriture du code que nous n'avons pas mis en place. Le développement web n'est pas compliqué et la structure MVC permet de comprendre facilement le code. De plus, les noms des fonctions et des fichiers ont été élaborés de manière à ce que le développement se comprenne comme du pseudo code.

Tests unitaires

Chaque classe de l'algorithme à l'exception de DataBase et main a été testée afin d'être sûr que l'algorithme fonctionnera pour toutes les élections différentes possibles.

Pour cela il a fallu tester pour chaque méthode publique des Classes si nous avions bien le code parcouru aux bonnes instructions et les bonnes valeurs de retour.

Nous avons utilisé le framework JUnit4 qui permet aisément d'obtenir les informations précédemment définies.

Les méthodes de l'algorithme ont été testées avec différentes valeurs permettant de traiter tous les cas possibles d'une élection. Il y a 3 cas distincts :

- Il y a un vainqueur de Condorcet qui est désigné comme vainqueur. Les tests vérifient que ce vainqueur est bien trouvé par l'algorithme ;
- Il y a un cycle dans le graphe et le vainqueur est désigné par une loi de probabilité sur les matchs gagnés des alternatives. Les tests vérifient que ces probabilités sont bien respectées ;
- Il n'y a pas de vainqueur et pas de cycle dans le graphe. Le vainqueur est désigné par une loi de probabilité uniforme sur les alternatives en tête. Les tests vérifient que ces probabilités sont bien respectées.

Tous les tests nous permettent d'affirmer que l'algorithme fonctionne dans tous les cas possibles. Cela permet de s'assurer que non seulement le livrable de l'algorithme est complet mais qu'en plus, s'il y a une incohérence dans la base de données, ce n'est pas dû à l'algorithme mais au site web et il sera donc beaucoup plus aisé de détecter l'erreur et de la corriger.

Tests d'intégration

Suite aux tests unitaires, nous avons effectué des tests d'intégration afin de vérifier que les différentes structures que nous avions créées communiquaient correctement entre elles.

En ce qui concerne les interactions entre l'algorithme et la base de données, les tests effectués ont consisté à rentrer manuellement dans la base de données via l'interface de phpMyAdmin plusieurs votes avec pour chacun de ces votes plusieurs alternatives. Ensuite plusieurs choix, publics et privés, ont été insérés dans la base de données de la même façon. Ces choix classant les alternatives des votes implémentés précédemment ont été sélectionnés si bien que le résultat que renvoie l'algorithme soit connu d'avance.

Les tests entre la base de données et l'algorithme ont été réalisés pas à pas. Le premier test a permis de vérifier que l'algorithme java peut se connecter à la base de données ; pour cela il suffit d'afficher « database connected » une fois que les étapes nécessaires à la connexion ont été effectuées. Le test suivant consiste à vérifier que l'algorithme arrive à lire correctement les données stockées dans la base de données. Cela se fait en affichant les différentes voix correspondant à l'identifiant d'un des votes rentrés préalablement dans la base de données. Il est alors aisé, en comparant à ce qui avait été rempli dans la base de données, de constater le bon fonctionnement de la connexion. Enfin, il est nécessaire de s'assurer que le programme java est capable d'inscrire dans la base de données l'alternative qui remporte le vote. Une fois encore ce test ne présente pas de difficulté grâce à l'outil phpMyAdmin qui donne directement accès au contenu de la base de données et donc aux alternatives gagnantes renvoyées par l'algorithme.

Tests de validation

Les tests de validations ont été effectués en créant un vote, puis en votant plusieurs fois à partir de comptes différents et en le clôturant. Le test a alors été considéré comme réussi par la plateforme s'il était possible d'avoir accès au résultat de ce vote via plusieurs comptes différents et si le résultat du vote était bien inscrit dans la base de données.

Réalisé des fonctionnalités

	Contraintes
Algorithme de vote	Condorcet randomisé
Plateforme	Site web
Vote public	-Créer un vote par un administrateur -Générer un lien partageable pour accéder au vote -Le votant vote
Vote privé	-Créer un vote par un administrateur ayant un compte -L'administrateur liste les personnes autorisées à voter -Générer un lien partageable pour accéder au vote -Notifier les votants de la création d'un vote sur leur compte et par mail -Le votant peut accéder au vote via son compte si autorisé -Le votant vote
Sécurisation	-Empêcher la récupération des données privées des utilisateurs et des votes -Empêcher un utilisateur de voter plusieurs fois en rafraichissant la page -Protéger la plateforme web de différentes attaques comme l'injection SQL -Blockchain (si le temps le permet)

Réalisé
Non réalisé

On voit que pour un vote privé il n'est pas possible que l'administrateur liste les personnes autorisées à voter. Néanmoins pour rendre utile la fonctionnalité de vote privé, nous avons décidé que le temps d'implémenter le code complet, seul les personnes connectées peuvent accéder aux votes privés.

Réalisé du cahier des charges

Nom de la page	Contenu de l'interface
Utilisateur non connecté	-barre de recherche pour trouver des votes publics -barre de connexion -bouton créer un nouveau compte -liste des votes répartis sur plusieurs pages accessibles en cliquant sur la numérotation des pages -onglet créer un vote

Utilisateur connecté	-barre de recherche pour trouver des votes publics -onglet contenant la liste des votes répartis sur plusieurs pages accessibles en cliquant sur la numérotation des pages -onglet contenant la liste des votes auxquels il a accès en spécifiant s'il n'a pas encore voté, s'il a voté mais que le vote n'est pas clôturé et si le vote est clôturé -onglet de déconnexion -bouton valider
Créer un compte	Zone de texte : -pseudo -adresse mail -mot de passe -confirmation du mot de passe -bouton valider
Créer un vote	-objet du vote -les différentes alternatives -cocher l'option vote privé ou vote public -choisir la date d'ouverture du vote -choisir la date de clôture du vote -bouton valider
Créer un vote public	-objet du vote -les différentes alternatives -choisir la date d'ouverture du vote -choisir la date de clôture du vote -bouton valider
Option vote privé	-formulaire à remplir avec les adresses des personnes autorisées à voter -affichage du lien du vote -bouton valider
Option vote public	-affichage du lien du vote -bouton valider
Voter	-affichage de l'objet du vote -pour chaque position dans le vote, le votant sélectionne l'alternative qu'il souhaite dans un menu déroulant -bouton valider
Accéder aux résultats d'un vote	-afficher l'objet du vote -afficher les résultats du vote sous forme d'un camembert

Réalisé

Non réalisé

Un utilisateur connecté n'a pas accès à l'information s'il a déjà voté sur un vote à moins qu'il essaye de voter ce qui lui donnera une erreur. Il peut donc tout de même voter seulement une fois.

Il est possible de choisir la date d'ouverture et clôture d'un vote, néanmoins ces informations sont inutiles à l'heure actuelle car le site web charge tous les votes peu importe leur date d'ouverture et ne clore pas les votes automatiquement à leur date de fin.

Manuel utilisateur

Pour le moment la plateforme web n'est disponible que localement. On y accède via n'importe quel navigateur web par le lien amenant à l'accueil.

Une fois sur cette page, un texte de présentation accompagné d'une vidéo nous explique le principe de la méthode de Condorcet.

En haut de la page, l'utilisateur a accès aux principales fonctionnalités du site : voir tous les votes, créer un vote, se connecter ou se déconnecter, s'inscrire. Les votes auxquels l'utilisateur a accès sont les votes publics et les votes privés s'il est connecté.

En bas de la page, l'utilisateur peut contacter un membre de l'équipe pour signaler un problème ou donner son avis.

Dans la page listant tous les votes, chaque vote apparait avec ses caractéristiques principales comme son type (public ou privé) et son statut (ouvert ou fermé) par exemple. L'utilisateur peut voter s'il est connecté ou déconnecté et si le vote est encore ouvert. Chaque utilisateur connecté peut voter une fois uniquement sur chaque vote. L'administrateur d'un vote (celui ou ceux qu'il a créé) peut voter à un vote qu'il a créé ou à d'autres votes créés par d'autres utilisateurs. Il peut aussi clôturer un vote qu'il a créé, auquel personne ne pourra voter ensuite. Pour les votes fermés, les utilisateurs peuvent voir le résultat du vote.

Pour créer un vote, l'utilisateur doit être connecté. Il faut d'abord choisir le nombre d'alternatives possibles. On accède ensuite à une autre page où l'on pourra définir les autres caractéristiques du vote. Le vote créé apparaîtra ensuite dans la liste de tous les votes.

Manuel Développeur

Il reste de nombreuses fonctionnalités sur le site web à mettre en place et ce guide a pour but d'aider à pouvoir les mettre en place.

Pour pouvoir ajouter de nouvelles fonctionnalités, le développeur va devoir utiliser la structure MVC introduite par notre développement. Il devra donc s'approprier les différents modèles et comprendre ce que représente chaque contrôleur et dans quel cas il devra en appeler un plutôt qu'un autre ou en créer un nouveau.

Par exemple pour implémenter la fonctionnalité permettant à l'administrateur d'un vote privé de choisir quelles personnes inscrites pourront voter, il devra créer une nouvelle vue permettant d'entrer l'email de ces personnes. Cela sera géré par controllerNouveauVote qui permet déjà de créer de nouveaux votes et dans lequel il faudra créer une nouvelle fonction. Le développeur devra également implémenter de nouvelles méthodes dans la classe VotantPrive du modèle qui représente la table du même nom, au pluriel près, dans la base de données afin de pouvoir ajouter ces votants autorisés.

L’algorithme ne devra pas être modifié. En effet, il a été testé afin de répondre au cahier des charges et fonctionne parfaitement. Le toucher serait ne plus répondre au besoin exprimé, ce qui serait donc une erreur.

Conclusion

Ainsi, dans ce rapport on a pu retrouver de nombreuses explications sur le déroulement de notre projet et sur notre plateforme de vote en ligne.

Des améliorations sont évidemment possibles sur ces points :

- Le design de l’interface, afin de la rendre plus esthétique et complète ;
- Un module de recherche d’un vote dans la liste de tous les votes ;
- Un module de tri des votes en fonction de leur statut ou de leur type dans la liste de tous les votes ;
- Le contenu des résultats, en affichant le nombre de matchs gagnés par chaque alternative par exemple ;
- L’utilisation des dates pour ouvrir et clôturer un vote ;
- …

Une dernière fonctionnalité dont on avait discuté aux prémices du projet était l’implémentation d’un système de blockchain pour les votes.

Bibliographie

Cours en ligne sur HTML et CSS : https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3#

Cours en ligne sur PHP et MYSQL : https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql

Cours en ligne sur les tests unitaires en Java : https://openclassrooms.com/courses/les-tests-unitaires-en-java

Cours en ligne sur Java EE : https://openclassrooms.com/courses/creez-votre-application-web-avec-java-ee

Informations sur l'utilisation de Bootstrap : https://getbootstrap.com/docs/3.3/css/

Gestion de projet

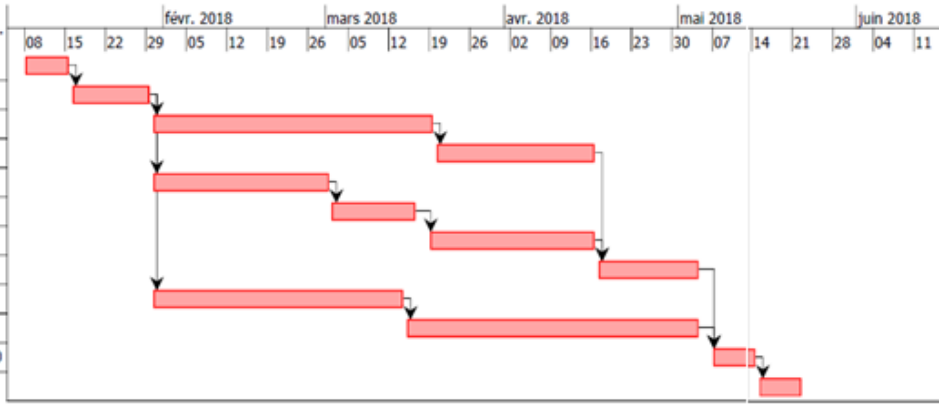
Plan de charge

Vote à la Condorcet randomisé	**PLAN DE CHARGES PRÉVISIONNEL**						**SUIVI D'ACTIVITÉS (Charge Consommée)**					
			Charge en H / Participant						**Charge en H / Participant**			
Description de l'activité	**Charge en %**	**Charge en H**	EC	JD	RM	TP	**Charge en %**	**Charge en H**	EC	JD	RM	TP
Total	**100%**	**209**	**41,5**	**64,5**	**53,5**	**49,5**	**89%**	**186,5**	**42,5**	**102**	**41,5**	**0,5**
Gestion de projets	**24%**	**50**	**12,5**	**12,5**	**12,5**	**12,5**	**21%**	**39,5**	**13,5**	**15**	**10,5**	**0,5**
Réunion de lancement	1%	2	0,5	0,5	0,5	0,5	1%	2	0,5	0,5	0,5	0,5
Planning prévisionnel et Suivi d'activités	2%	4	1	1	1	1	2%	3	1	1	1	0
Réunions de suivi	11%	24	6	6	6	6	5%	10	3	4	3	0
Rédaction du rapport	8%	16	4	4	4	4	11%	20	8	7	5	0
Outils collaboratifs (svn, etc.)	2%	4	1	1	1	1	2%	4,5	1	2,5	1	0
Spécification	**4%**	**8**	**2**	**2**	**2**	**2**	**4%**	**8**	**2**	**3**	**3**	**0**
Définition des fonctionnalités	4%	8	2	2	2	2	4%	8	2	3	3	0
Conception préliminaire	**4%**	**8**	**2**	**2**	**2**	**2**	**3%**	**6**	**2**	**2**	**2**	**0**
Définition d'un format de fichiers et d'un modèle de données	4%	8	2	2	2	2	3%	6	2	2	2	0
Conception détaillée	**22%**	**45**	**9**	**12**	**14**	**10**	**21%**	**39,5**	**11**	**15,5**	**13**	**0**
Définition des classes	1%	3	0	1	1	1	2%	3,5	0	2,5	1	0
Définiton des méthodes	1%	2	0	1	1	0	2%	4	0	3	1	0
Définition des tests unitaires	5%	10	1	3	3	3	1%	2	0	2	0	0

Auto-formation	8%	16	5	4	5	2	10%	19	9	3	7	0
Préparation de la base de données	3%	7	2	1	2	2	4%	7	2	1	4	0
Structure MVC	3%	7	1	2	2	2	2%	4	0	4	0	0
Codage	**41%**	**86**	**13**	**33**	**20**	**20**	**44%**	**81,5**	**11**	**62,5**	**8**	**0**
Codage des classes	1%	3	0	1	1	1	8%	15,5	0	14,5	1	0
Codage des méthodes	10%	21	0	6	6	6	5%	9	0	5	4	0
Codage de l'interface web	10%	21	6	6	6	6	6%	12	10	2	0	0
Codage des tests unitaires	6%	12	3	3	3	3	5%	10	0	10	0	0
Sécurisation de l'appplication web	4%	9	3	2	2	2	1%	1,5	0	1,5	0	0
Commentaires de code	0%	0	0	0	0	0	4%	7	1	5	1	
Codage MVC	10%	20	1	15	2	2	14%	26,5	0	24,5	2	0
Intégration	**6%**	**12**	**3**	**3**	**3**	**3**	**6%**	**12**	**3**	**4**	**5**	**0**
Intégration des modules	4%	8	2	2	2	2	4%	7	2	2	3	0
Tests d'intégration	2%	4	1	1	1	1	3%	5	1	2	2	0

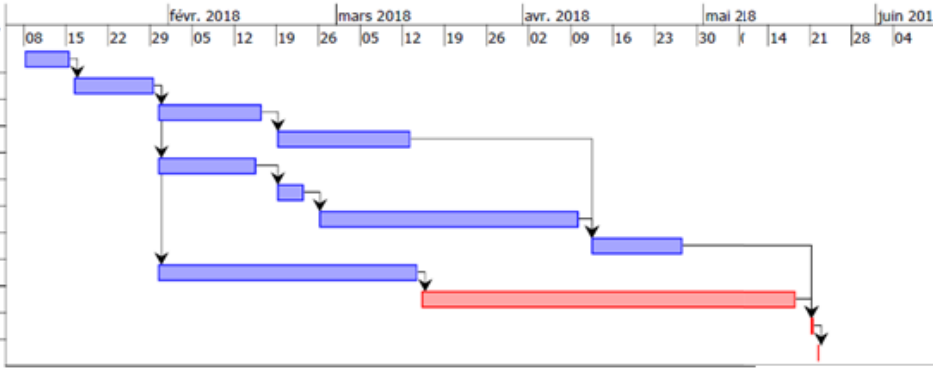
Planning prévisionnel

	Nom	Durée	Début	Fin	Pré...
1	Définition des fonctionnalités	6 jo...	08/01/18 ...	15/01/18 ...	
2	Format de fichier et d'un modèle de données	10 j...	16/01/18 ...	29/01/18 ...	1
3	Algorithme java	35 j...	30/01/18 ...	19/03/18 ...	2
4	Tests unitaires de l'algorithme	20 j...	20/03/18 ...	16/04/18 ...	3
5	Conception de la base de données	23 j...	30/01/18 ...	01/03/18 ...	2
6	Préparation de la base de données	11 j...	02/03/18 ...	16/03/18 ...	5
7	Intégration de l'algorithme à la bdd	21 j...	19/03/18 ...	16/04/18 ...	6
8	Tests d'intégration	14 j...	17/04/18 ...	04/05/18 ...	4;7
9	Codage des vue du site web	32 j...	30/01/18 ...	14/03/18 ...	2
10	Structure MVC	37 j...	15/03/18 ...	04/05/18 ...	9
11	Intégration du site avec l'algorithme	6 jo...	07/05/18 ...	14/05/18 ...	8;10
12	Rédaction du rapport final	6 jours	15/05/18 ...	22/05/18 ...	11



Planning réalisé

	Nom	Durée	Début	Fin	Préd...
1	Définition des fonctionnalités	6 jo...	08/01/18 ...	15/01/18 ...	
2	Format de fichier et d'un modèle de données	10 j...	16/01/18 ...	29/01/18 ...	1
3	Algorithme java	14 j...	30/01/18 ...	16/02/18 ...	2
4	Tests unitaires de l'algorithme	16,...	19/02/18 ...	13/03/18 ...	3
5	Conception de la base de données	13 j...	30/01/18 ...	15/02/18 ...	2
6	Préparation de la base de données	5 jo...	19/02/18 ...	23/02/18 ...	5
7	Intégration de l'algorithme à la bdd	31,...	26/02/18 ...	10/04/18 ...	6
8	Tests d'intégration	11,...	12/04/18 ...	27/04/18 ...	4;7
9	Codage des vue du site web	32 j...	30/01/18 ...	14/03/18 ...	2
10	Structure MVC	47 j...	15/03/18 ...	18/05/18 ...	9
11	Intégration du site avec l'algorithme	1 jo...	21/05/18 ...	21/05/18 ...	10;8
12	Rédaction du rapport final	1 jour	22/05/18 ...	22/05/18 ...	11



Suivi d'activités

Après avoir constitué l'équipe, nous avons commencé à chercher, chacun de notre côté, des propositions de sujets. Nous nous sommes ensuite réunis pour confronter les idées de chacun et choisir parmi elles, celles qui présentait le plus d'intérêt tant d'un point de vu de la technicité que de la faisabilité ou de la diversité des technologies employés. Une fois le sujet de plateforme de vote à la Condorcet randomisé fut validé par notre professeur référent, nous avons effectué plusieurs réunions afin de développer tous ensemble l'idée de départ et de déterminer les différentes fonctionnalités à donner à la plateforme. Ces réunions, au nombre de 5 ont été physiques puis par vidéo-conférence à l'aide de l'outil Hangouts de Google. Elles nous ont permis dans un premier temps de nous mettre d'accord sur le cahier des charges à mettre en œuvre dans ce projet. Ensuite nous avons déterminé les différents langages que nous allons utiliser. Enfin, nous avons commencé à écrire le code tous ensemble en définissant les principales classes qui vont composer les différentes structures (site web, base de données et algorithme). Pour chacune des réunions que nous faisions, un compte-rendu était rédigé et partagé sur le drive réservé à ce projet.

Face à la difficulté grandissante d'écrire un algorithme à plusieurs, nous nous sommes réparti les rôles. Nous avons développé individuellement la partie du code qui nous était attribuée. Pour nous

tenir au courant de l'avancée de chacun en temps réel, nous avons utilisé l'outil Trello. Ainsi nous disposions de listes de tâches catégorisées en fonction de leur état d'avancement (to do, doing, done). Ces tâches étaient attribuées à un ou plusieurs membres du groupe qui pouvaient alors les diviser en lots de sous-tâches qu'ils validaient selon la progression dans le projet. Les membres du projet devaient aussi commenter chaque tâche qu'ils effectuaient en indiquant le temps passé dessus. Notre Trello possédait également une liste help où nous pouvions poster des cartes de questions à destination de notre professeur référent.

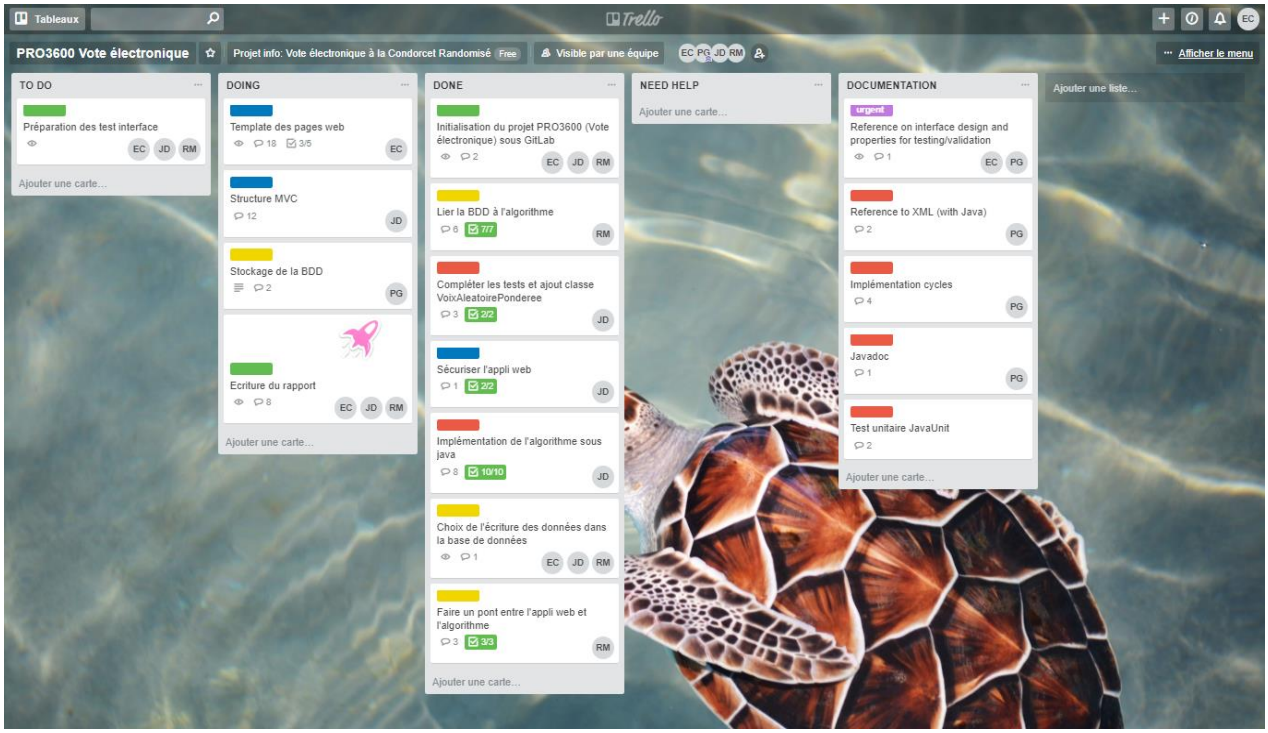


Tableau de suivi du projet avec Trello

Un autre outil qui nous a été utile est Gitlab, un logiciel de gestion de versions décentralisées qui nous permettait d'accéder directement au code de chacun de nous. Nous avons tout particulièrement eu recours à Gitlab afin de partager les vues du site web pour qu'elles soient ensuite intégrées à l'architecture MVC. Nous l'avons aussi utilisé lors de la phase d'intégration en particulier pour récupérer les structures codées par des personnes différentes afin de les relier entre elles.

En ce qui concerne nos communications plus informelles entre membres du groupe, nous nous sommes servis de Messenger. Quant à nos échanges avec notre professeur référent, principalement dans le but de fixer des rendez-vous, elles ont eu lieu par mail.

Pour la phase de finalisation de notre projet, avant de le rendre, pour optimiser l'efficacité de nos échanges, nous nous sommes réunis et avons procéder aux derniers ajustements et à la rédaction du rapport lors de séances de travail collectives. Le rapport a été rédigé sur un fichier partagé sur le drive où chacun complétait la partie qui lui a été attribuée.

Ainsi, ce projet informatique a constitué un véritable travail de groupe entre ses différents acteurs. Nous avons employé divers moyens de communication, chacun associé à une fonction précise (partager du code, des informations, demander de l'aide…)