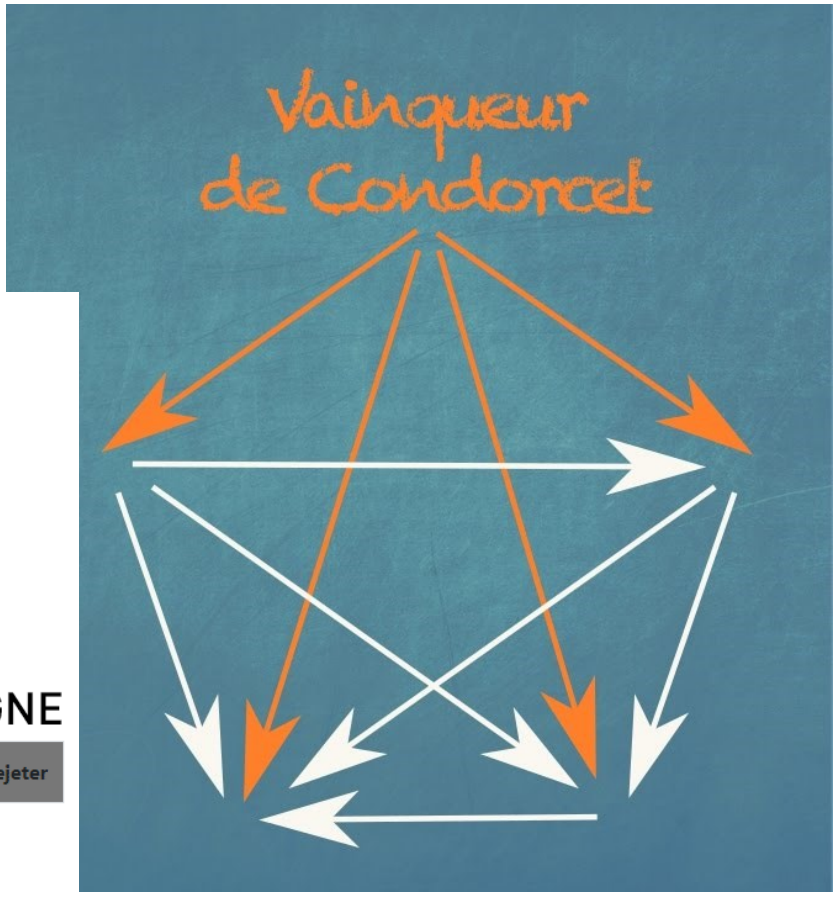

U.E. 2.4: « Projet Informatique »

Rapport de projet: Des votes alternatifs

01 juin 2022

ENSTA BRETAGNE

2 Rue François Verny
29806 Brest Cedex 9, France

Simpore Sheick, *sheick.simpore@ensta-bretagne.org*
Lemaitre Jérémy, *jeremy.lemaitre@ensta-bretagne.org*

FISE2024, TD5

Table des matières

Table des figures

Chapitre 1

Description générale du projet

1.1 Introduction

Au XXI siècle, nous vivons dans un monde marqué par un courant démocratie. Dans les différents pays du globe, chaque gouvernement est représenté au travers un vote au suffrage universel. De nos jours, pour rendre ces votes efficients, l'informatique peut apporter sa contribution par la création d'environnements afin de faciliter la réalisation de ces différentes élections. De plus, en cette année d'élection, on peut s'interroger sur la pertinence du scrutin uninominal à deux tours. En effet le marquis de Condorcet mathématicien et philosophe du XVIIIème siècle énonça le principe suivant : « si une alternative est préférée à tout autre par une majorité, alors cette alternative doit être élue ». Ainsi l'alternative élue devrait pouvoir remporter tout duel face à une autre alternative. Des études statistiques montrent que ce ne serait pas toujours le cas en politique et que le mode de scrutin y est pour beaucoup.Or de nombreux modes de scrutins alternatifs existent (scrutin de Condorcet randomisé, jugement majoritaire, méthode de Borda, méthode de Coombs, ...). Bien que ces derniers ne soient pas parfaits, semblent mieux répondre au principe énoncé par Nicolas de Condorcet. Ces méthodes étant difficiles à mettre en œuvre à grande échelle, leur utilisation reste souvent limitée. C'est ici l'informatique pourrait-être un grande aide. À ce jour, plusieurs applications mobiles permettent de mettre en oeuvre des élections, mais aucune ne nous a apporté une entière satisfaction. Ce projet est donc l'occasion pour essayer de développer une application, nommée "Votes Alternatifs", répondant à nos attentes, et contribuer au développement du vote informatisé. Ainsi, l'objectif de ce projet serait de créer une application simple d'utilisation et qui pourrait être utilisée pour réaliser des votes au sein d'un groupe lorsque le nombre d'alternatives est supérieur à 3.

1.2 Les pistes envisagées

Pour cadrer le projets, quelques exigences, nous semblant indispensable au bon déroulement du vote, ont été définies. Tout d'abord pour pouvoir voter il faut que l'élection existe. C'est pourquoi la personne ou l'entité en charge du vote doit pouvoir grâce à l'application :

— Créer une élection en lui donnant un nom.

— Définir sur qu'elle durer le vote aura lieu.

— Choisir le mode de scrutin qui souhaite.

— Décider des personnes qui seront autorisées à voter.

— Inviter les électeurs à participer au vote

De plus, dans la mesure où une élection est sensible et généralement bien cadrée, il est nécessaire de s'assurer de la sécurité et de la confidentialité au sein de l'application. C'est-à-dire :

— S'assurer que les personnes qui votes sont autorisées à le faire.

— S'assurer que chaque personne ne vote pas plus de deux fois.

— Garantir l'anonymisation des bulletins de vote.

— S'assurer que l'élection n'est pas falsifiable.

Enfin, l'administrateur de l'élection, ainsi que les électeurs, devraient pouvoir avoir accès au résultats de façon claire et totalement transparente.

Pour répondre à ces nombreux objectifs, que nous n'avons malheureusement pas tous remplis, plusieurs pistes ont été envisagées.

1.3 Hypothèses et simplifications

Dans la mesure où le travail à réaliser était important et le temps était limité, nous avons décider de réaliser quelques simplifications et hypothèses pour pouvoir mener à bien le projet.

Tout d'abord, nous avons décider de nous limiter à deux modes scrutins alternatifs : le scrutin de Condorcet et le Jugement Majoritaire. Ces modes de scrutin étaient à la fois ceux qui nous connaissions le mieux et ceux qu'il nous semblaient le plus intéressant d'implémenter dans une telle application. Pour coder ces deux modes de scrutins, nous nous sommes inspiré de deux programmes qui existaient déjà :

— Condorcet : *Condorcet: Implement multiple-round condorcet elections*, Radii (Github)

— Jugement Majoritaire : *majorityVote*, Shatoshishi00 (Github)

Ensuite nous avons considérer que les électeurs votaient chacun leur tour, sur le même ordinateur. En réalité ceci ne pourrait pas être envisageable pour une élection à grande échelle car le temps requis pour que tous les électeurs votent serait trop important.

Par ailleurs, notre application permet essentiellement de faire des votes pour des candidats humains qui sont eux-mêmes électeurs. Dans les fait, une telle application pourrait être utilisée dans de nombreux autres cas comme par exemple, au sein d'un groupe d'amis, faire le choix de l'activité du week-end.

Enfin, nous avons fait l'hypothèse que dans tous les cas, un et un seul vainqueur serait désigné par ces modes de scrutin. En pratique des cas d'égalité existent mais la probabilité qu'ils surviennent est très faible à partir du moment où le nombre d'électeur est suffisamment élevée. Dans les fait si cela arrivait, certaines personne proposent de tirer le vainqueur.

Chapitre 2

Mise en oeuvre de l'application

2.1 Présentation générale du programme

2.1.1 Le diagramme des classes

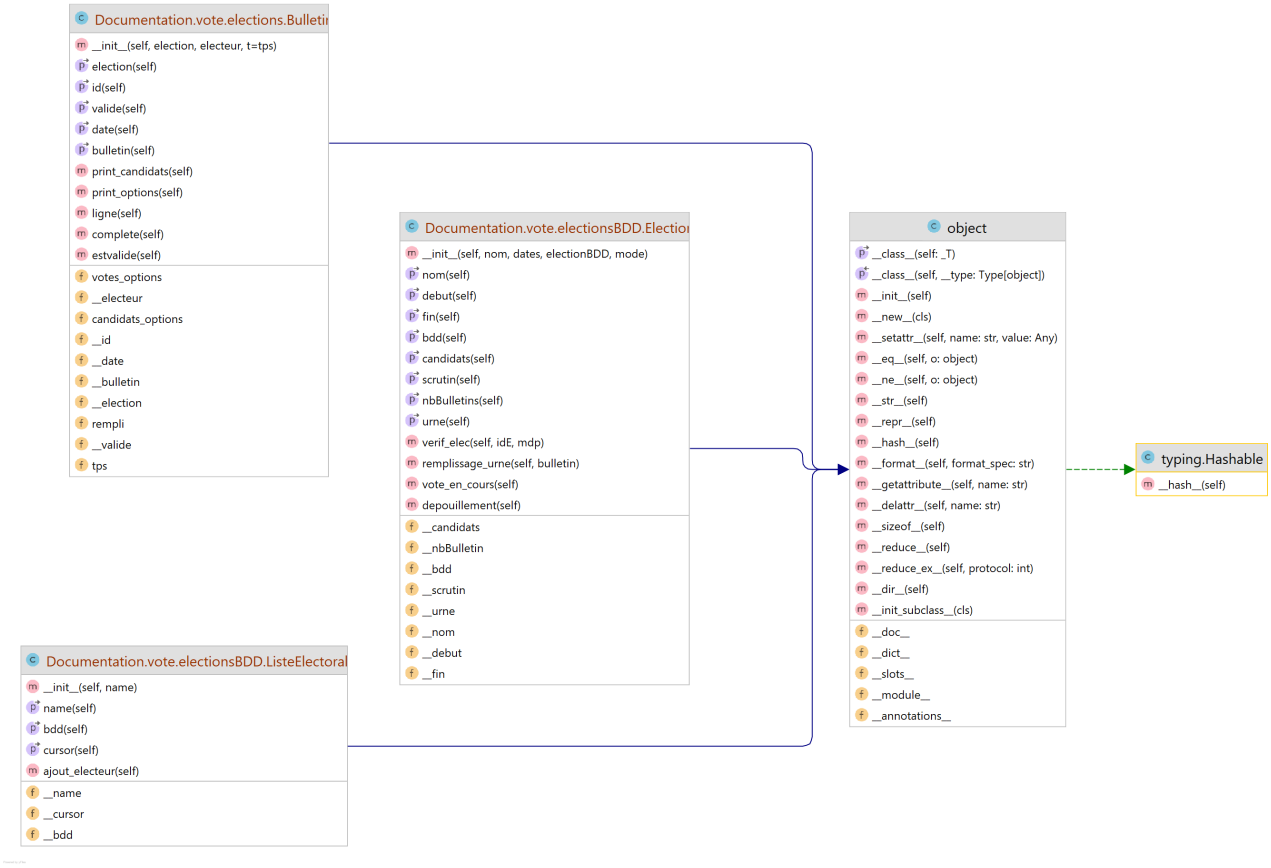

FIGURE 2.1 – Digramme des classes du module *electionsBDD*

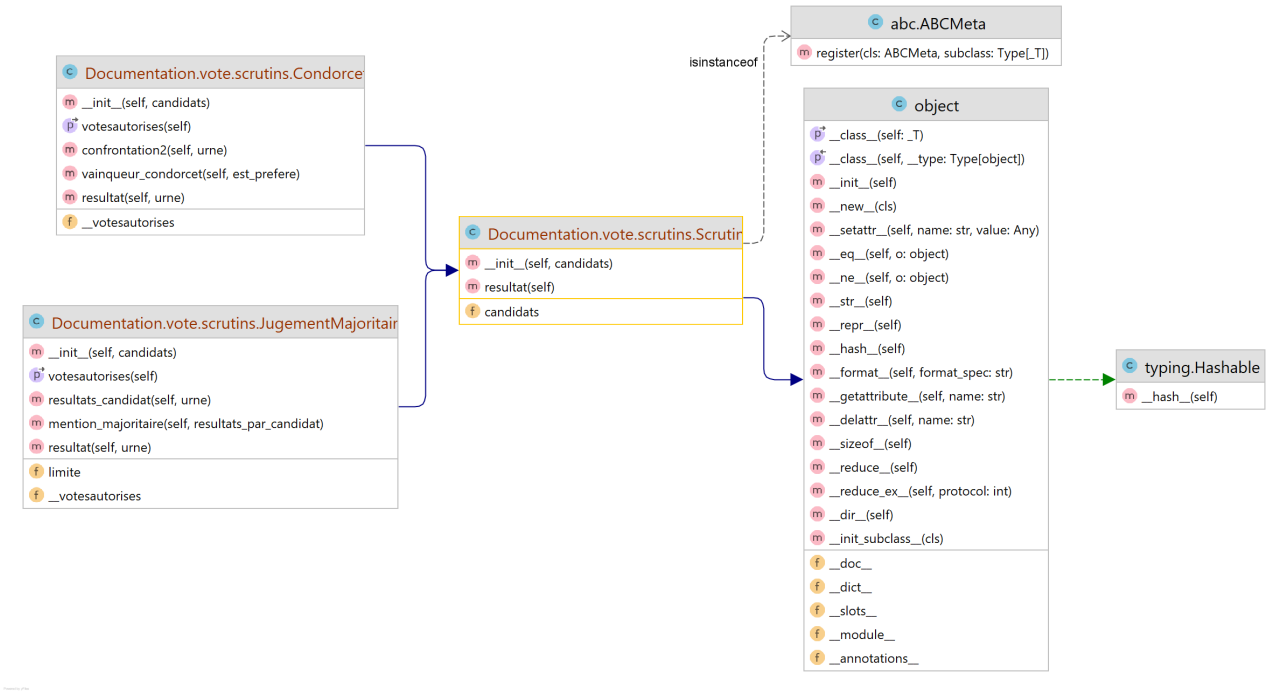

FIGURE 2.2 – Digramme des classes du module *scrutins*

2.2 Les modules et les classes

2.2.1 Le module *electionsBDD*

Afin de pouvoir créer une élection, il nous a semblé pertinent de regrouper tous les outils nécessaires au sein d'un même module appelé *electionBDD*. Ce module est composé de trois classes représentant chacune trois outils différents :

2.2.1.1 La classe *Election*

La classe *Election* correspond à l'élection générale qui possède un nom, une date et heure de début et de fin, des candidats, des électeurs et des règles qui correspondent ici au mode de scrutin.

2.2.1.2 La classe *Bulletin*

La classe *Bulletin* correspond au vote d'un unique électeur. Dans ce bulletin est renseigné, sous la forme d'un dictionnaire, le nom du candidat choisi et la mention qui lui est attribuée. Les mentions dépendent bien sûr du mode de scrutin qui est choisi. La méthode **.estvalide()** de cette classe permet de vérifier que le bulletin est bien valide, c'est-à-dire :

1. que la date et l'heure à laquelle le bulletin est déposé appartienne bien à la période de vote.
2. que l'électeur soit bien présent dans la liste électorale, qu'il soit bien autorisé à voter et qu'il n'est pas déjà voté.
3. que les candidats renseignés appartiennent à la liste des candidats.
4. que les votes soumis appartiennent bien aux votes autorisés par le scrutin.

Une fois que ce bulletin est validé, il peut être déposé dans l'urne via la méthode de la classe *Election*, **.remplissage_urne()**

2.2.1.3 La classe *ListeElectorale*

La classe *ListeElectorale* permet d'intéragir avec une base de donnée pour créer une base de donnée et la remplir ainsi que pour récupérer les éléments utiles dans nos autres classes. Cette classe n'est pas totalement a point. Nous présenterons la gestion des bases de données en Partie **??**.

2.2.2 Le module *scrutins*

Ce module comporte tout le calcul mathématique de notre projet. Il présente trois classes (scrutins, jugement majoritaire, condorcet).

2.2.2.1 La classe mère *Scrutin*

C'est la classe mère du module. C'est une classe abstraite contenant deux méthodes.

1. En premier lieu il y a le constructeur menu d'une instanciation, permettant de créer l'objet candidat sous forme de liste de candidat. second lieu, il y a la méthode abstraite résultat. Les classes filles (jugement majoritaire et condorcet) se serviront de cette méthode pour retourner le candidat gagnant.

2.2.2.2 La classe fille *Condorcet*

1. La classe Condorcet hérite de scrutin.
2. Elle comporte cinq méthode décrivant le calcul mathématiques décrivant le calcul mathématique permettant de déterminer le vainqueur par condorcet à partir des différents vote.

2.2.2.3 La classe fille *JugementMajoritaire*

1. Tout comme la classe Condorcet, elle hérite de scrutin
2. Génère le calcul mathématique qui permet de déterminer le vainqueur par jugement majoritaire.

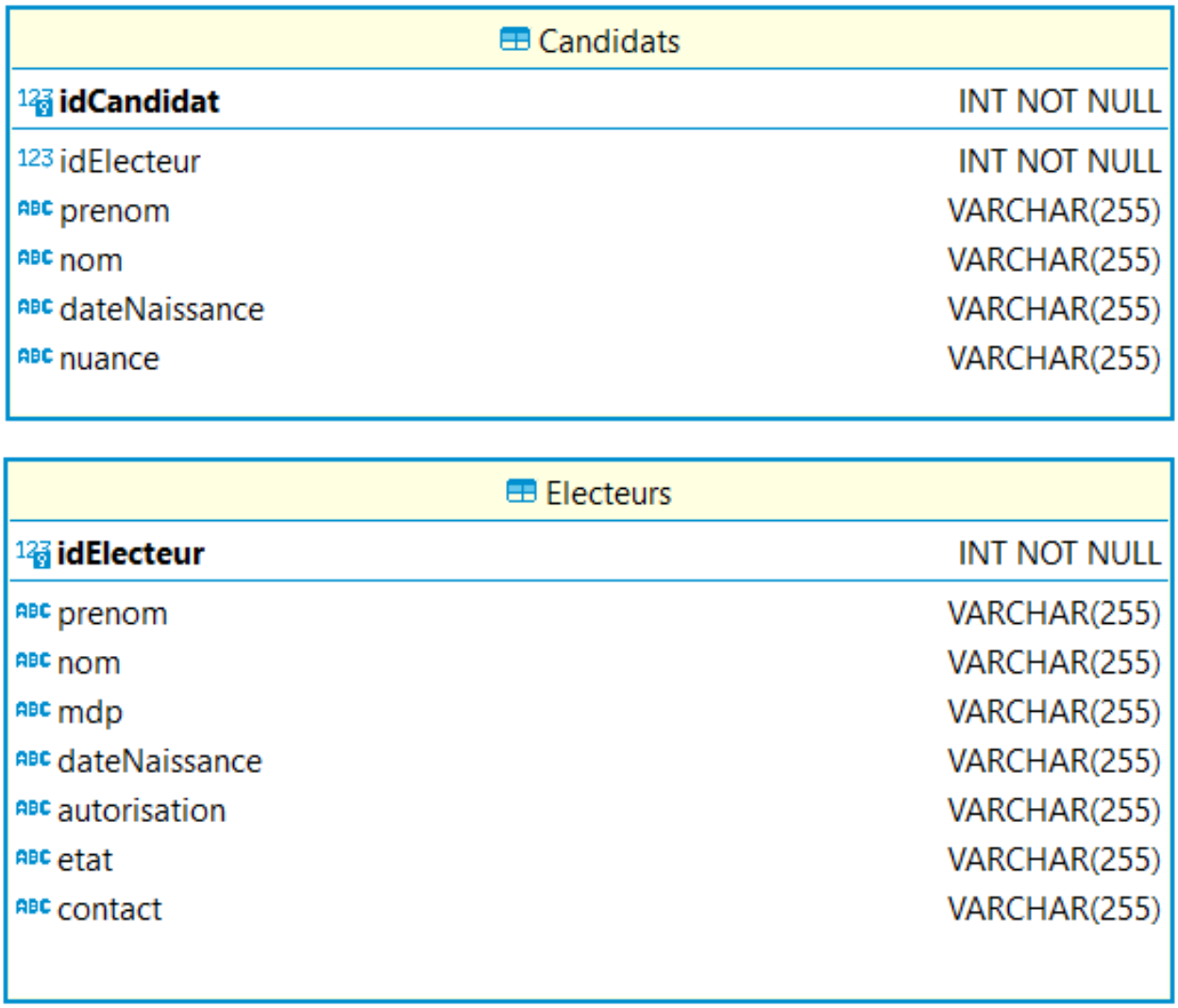

FIGURE 2.3 – Structure des bases de données utilisées dans l'application

2.3 Gestion de bases de données

Comme rappelé précédemment, l'objectif de cette base de données est de regrouper les informations sur les candidats et les électeurs pouvant participer au vote. Ainsi les bases de données que nous utilisons sont composées de deux tables : *Electeurs* et *Candidats*. Ci-après, la structure type d'une base de données utilisée pour l'élection (Figure 2.3). Dans la table *Electeurs*, l'attribut "mdp" correspond au mot de passe de l'électeur, qui est générée aléatoirement grâce à la fonction "generate_mdp(longueur mot de passe)", l'attribut "autorisation" vaut 1 si l'électeur à la permission de voter, l'attribut "etat" indique 0 si l'électeur n'a pas encore voté et enfin l'attribut "contact" renseigne l'adresse email de l'électeur afin de pouvoir lui envoyer son identifiant et son mot de passe pour l'élection.
Afin de manipuler les données de la base de données nous avons fait le choix, par simplicité, de rester en local. Pour cela nous avons utilisé le langage SQL via la bibliothèque python sqlite3.

2.4 L'Interface Homme-Machine

L'IHM a été conçu avec l'interface graphique pyQt5.Elle comporte d'abord une première page, qui permet à l'utilisateur de créer une élection ou de voter ou d'afficher les résultats de l'élection.

Chapitre 3

Conclusion

3.1 Le respect des figures

Le code que nous avions établi respecte les sept figures imposées. Le tableau ci-dessous explique ces sept figures en montrant certaines parties du codes ou ils interviennent.

Figures imposées	Respecté	Commentaires
1. Factorisation du code	Oui	Notre code comporte en tout trois modules à savoir le module Election, le module scrutin et le module ElectionsBDD. Dans chaque module les classes qui y existent sont différents tant par le nom que par le code.
2. Documentation et commentaires de code	Oui	Chaque classe de chaque module comporte des docsrting propre a lui-même qui explique le bout de code concerné. A partir de cela nous avions créé une documentation(voir annexe) avec la bibliothèque Sphinx.
3. Test unitaires	Oui	Nous avions réalisé des tests unitaires pour chaque module cité plus haut.
4. Création d'un type d'objet	Oui	Dans le module electionsBDD par exemple, les différentes classes possèdent plus d'une
5. Héritage	Oui	Cela est visite à travers les classes mères et classes filles que nous avions établi. Par exemple dans le module scrutins, la classe Condorcet et la classe JujementMajoritaire hérite de la classe mère scrutins
6. Accès base de donnée	Oui	Pour pouvoir stocker les données liées aux identifiants, les mots de passe des électeurs ainsi que les données liées aux candidats qui se présentent, nous avons créé une base de donnée (ElectionTest) qui s'occupe de cela en utilise le serveur sqlite3.
7. Lecture, écriture de fichiers	Oui	Par exemple, il est possible d'importer une autre base de données hormis celle que nous avions établie.

3.2 Les tests unitaires

Nous avons réalisé des tests unitaires pour chaque classe créée. On retrouve à chaque fois un test d'initialisation de la classe via la création d'une instance. Pour la classe Election est testé le bon remplissage de l'urne ainsi que la vérification de l'autorisation à voter des candidats. Pour la classe Bulletin est testé le remplissage du dictionnaire correspondant au bulletin et la vérification de

la validité du bulletin. Les tests sur les classes Condorcet et JugementMajoritaire correspondent eux majoritairement à des tests d'égalité.

3.3 Les limites de l'application

Malheureusement après deux mois de travail, l'application est loin d'être aboutie. En effet de nombreux objectifs initiaux ne sont satisfaits. En particulier :

— Nous ne sommes pas parvenu à implémenter l'envoi automatique de mails aux électeurs.

— Nous n'avons pas eu le temps d'approfondir la partie sécurité de l'application. En particulier, la confidentialité des bulletins laisse à désirer et aucun protocole n'a été mis en place pour s'assurer que les votes ne peuvent pas être falsifiés.

— L'affichage du résultat consiste seulement en l'affichage du nom du vainqueur. Nous n'affichons ni classement ni statistiques dans nos résultats.

Hormis ces objectifs non respectés, notre application possède de nombreux problèmes importants :

— Les votes pouvant être organisés à partir de l'application sont uniquement des votes impliquant des humains.

— L'application ne permet pas à l'heure actuelle de sauvegarder des élections et donc ne permet pas le déroulement de plusieurs élections simultanément.

— L'administrateur n'est pas le seul à pouvoir créer une élection.

— L'application ne permet pas de créer sa propre base de données.

— Seul 12 candidats peuvent être affichés sur l'IHM au moment du vote.

— L'application ne permet le vote à distance.

— ...

En résumé, notre application nécessiterai encore de très nombreuses améliorations et corrections.

3.4 Perspectives et améliorations

Comme souligné dans la Partie 3.3, de nombreux problèmes font que notre application est aujourd'hui à peine fonctionnelle. Pour l'améliorer il faudrait d'abord commencer par traiter les différents points selon leur ordre d'importance. Tout d'abord, il serait nécessaire de pouvoir d'une part sauvegarder l'élection, par écriture dans un fichier .txt par exemple, et d'autre part permettre à l'administrateur de créer facilement sa propre base de données. Ensuite, il faudrait pouvoir rendre l'application plus universelle en ne limitant pas les choix à des candidats humains et bloquer l'accès à la modification aux seuls administrateurs. Enfin une fois que tous ces problèmes seront réglées, on pourra regarder de plus près les questions de confidentialité, de vote à distance et d'envoi des informations par email. La partie traitement statistique des résultats ne serait a traité qu'une fois tous les problèmes précédents réglés.

Voir documentation aux fichier suivant :