UNIVERSITATEA TEHNICĂ A MOLDOVEI

Course Work

APPOO

Analysis, Design and Programming of a Voting System

FAF 141 student:
Ana-Maria Brinza

Supervisor :
Andrei Postaru

Chisinau 2017

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a general description about an election software package, which allows registered students.professors to cast their ballot during the election period in an university and to show use case diagrams, activity, deployment, components, sequence diagrams and others. In addition, this document will provide useful information about how we will be handling different scenarios. It is intended for the university administration, election commissioner, student population and the candidate.

1.2 Scope

This election software package will be able to handle verifying the student's ability to vote in the election, creating ballots of various types of elections, accepting the votes, reporting the votes, interfacing with candidates, university administration, student organizations, and anyone who desires information from the system. The various types of election include student government, Miss or Mister University, favorite teacher and advisor, and various constitutional amendments. The election system will be completely online, allowing students to vote from virtually anywhere. Each student will only be allowed one vote and the system will be able to produce both the election tallies and statistical summaries of voter turnout.

At the moment we don't have such a system in our university and all the voting that we have are time consuming. This system aims to ease the student life.

1.3 Definitions/Abbreviations

1. **Election:** A formal and organized process of being selected by vote
2. **Ballot:** A device used to cast votes in an election
3. **Poll:** Where the voting takes place during an election
4. **Candidate/Option:** A person/option that is nominated for an election
5. **EC**: Election Commissioner - a person that have the rights to create an election. (usually someone from university staff or from student organization)

2 System Analysis and Modeling

In this work I build an independent product that is affiliated with Technical University. Each student of the university is meant to have a unique id once logging into the portal. The languages that are able to run on the server are Javascript and mainly Python (I used Django Framework). Once a user login to the system using their unique id they enter the system. Once into the system, the user will have the ability to vote in the open polls.

For a better understanding and in order to ease the development procedure first I modeled the system (using UML diagrams), build some scenarios. All these will be presented in the next subsections.

2.1 User Characteristics

The potential users of the systems are mainly students at Technical University of Moldova ranging from the freshman level to graduate and/or professional level. Also, there are professors and administration stuff of the university. Each of them have specific roles in our system. We will illustrate each of them in the Use Case Diagram. Then based on Scenarios we will analyze and build Sequence Diagram and Activity Diagram.

2.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

In our case we have 3 main users: End User (which is student, professors and each user that has the voting rights), Election Commissioner and Administration that have more permissions. Also, we will have the Admin being behind all that: creating the EC and Administration.

2.2.1 Scenarios for the Use Case

In this section we will list more scenarios based on our Use Case Diagram

EC/Administration Create an Election

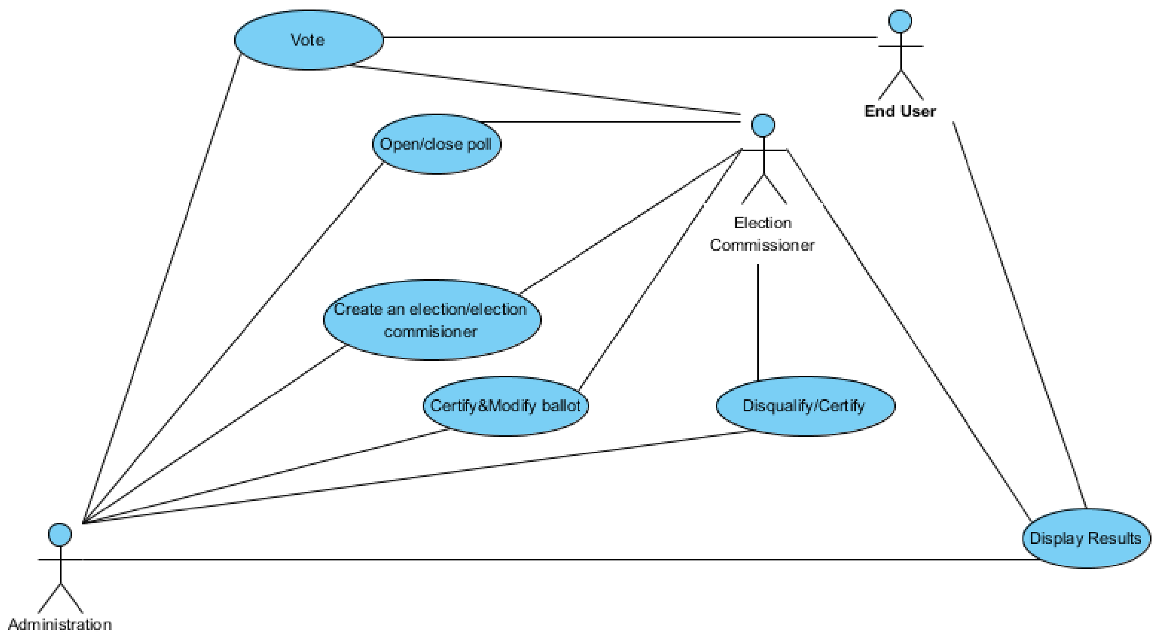


Figure 1: Use Case Diagram for the voting system

- **Use Case:** Create an Election
- **Iteration:** 1.0
- **Primary Actor:** Administration
- **Goal in Context:** To decide on the creation of a new election.
- **PreConditions:** Must hold an election
- **Trigger:** The election is held.
- **Scenario:**
  1. EC/Administration logs in
  2. EC/Administration clicks “Create Election” tab
  3. Present with “Create Election” page
  4. EC/Administration clicks adds an elections commissioner
  5. Present with election commissioner information into the dialog

6. EC/Administration enters election commissioner ID number into the dialog
7. EC/Administration assigns an election commissioner to the election
8. EC/Administration clicks “Create”
9. Election Commissioner is assigned and Election is created

- **Exceptions:**

1. No Election is required to held
2. There are no Election commissioners available
3. Duplicate Election information

- **Priority:** Essential
- **When Available:** 1.0
- **Frequency of Use:** For each case of Election
- **Channel to Actor:** Web browser
- **Secondary actor:** Admin
- **Channels to Secondary Actors**: Web browser
- **Open Issues:** N/A
- **Acceptance Tests:**

1. There is an even which needs an election and the election has been created by EC/Administration

a. The election will be added to Morgantown University Election System

2. There is an event with an election which has been created

a. The System already has the election

Students: Vote in Elections

- Use Case: students will have to vote eventually using the system
- Iteration: 1

- Primary Actor: students
- Goal in Context: To cast a vote
- Pre[U+2010]Conditions: Election must be created, ballot must be created and closed
- Trigger: Student selects a ballot, selects candidate, and confirms vote
- Scenario:
    1. Student logs in
    2. Check student eligibility on AR server
    3. Present student with "Election" page
    4. Student selects desired ballot
    5. Present student with ballot modal
        a. Modal contains one or more races
    6. Student selects desired candidate for each race in the ballot
    7. Student clicks "Vote"
    8. Submit vote to server
    9. Update records
- Exceptions:

1. Incomplete ballot
2. Ballot is open
3. Student is ineligible

- Priority: Essential
- When Available:1
- Frequency of Use: High use during election period
- Channel to Actor: AR, OIT
- Secondary actor: OIT, elections commissioner, AR, Head of student orgs

- Channels to Secondary Actors: API
- Open Issues: N/A
- Acceptance Tests:

1. Student user logs into system, selects a ballot on which he or she is eligible to vote, votes appropriately, and confirms their vote

a. Vote is cast, vote information is saved to the ballot

2. Student user logs into system and there are no ballots for which he or she is eligible to vote

a. No ballots appear for the user

2.3 Sequence Diagrams

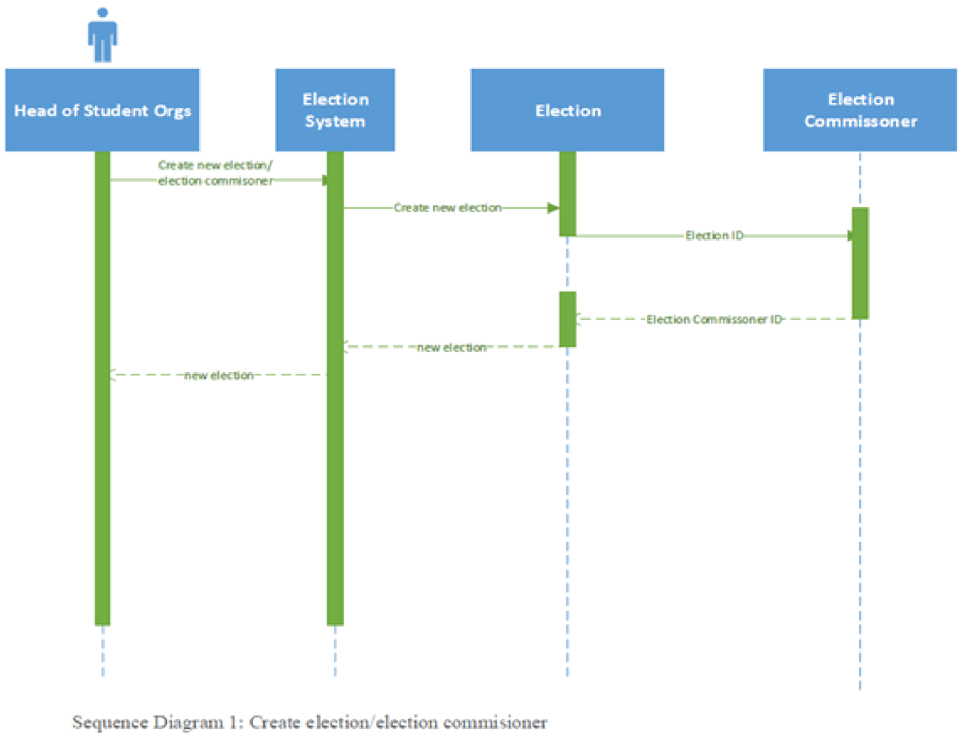
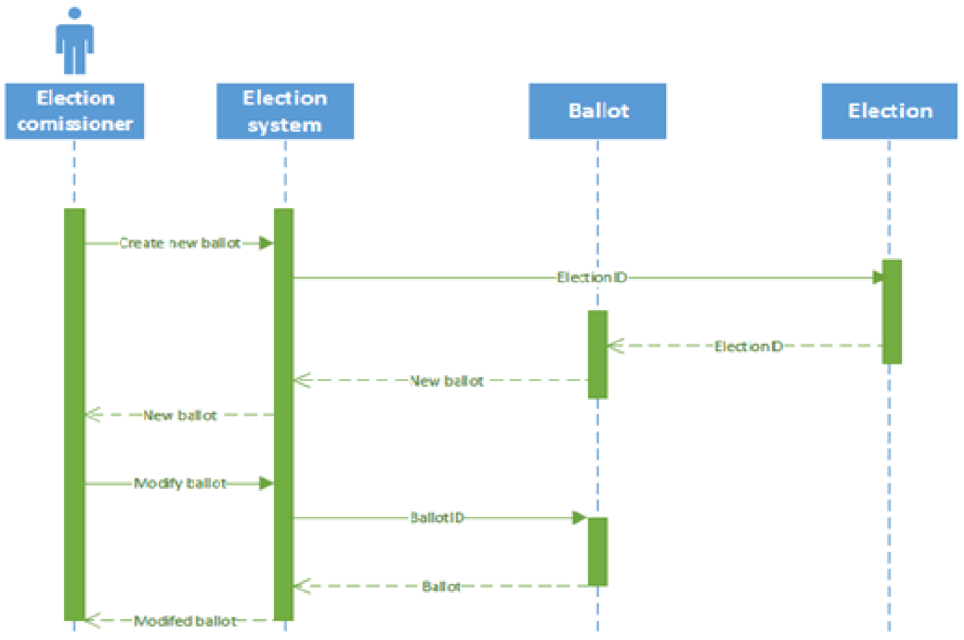
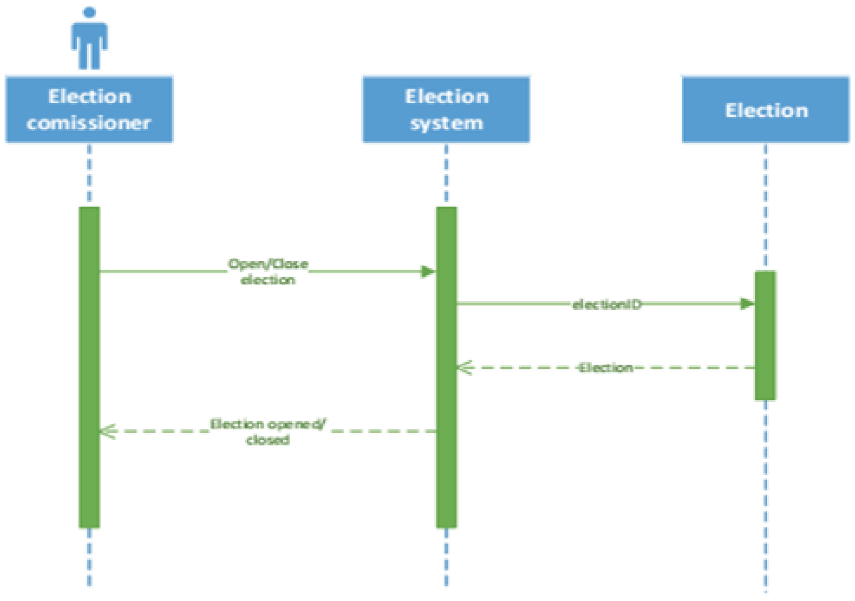


Figure 2: Sequence Diagram: Create election/election commissioner



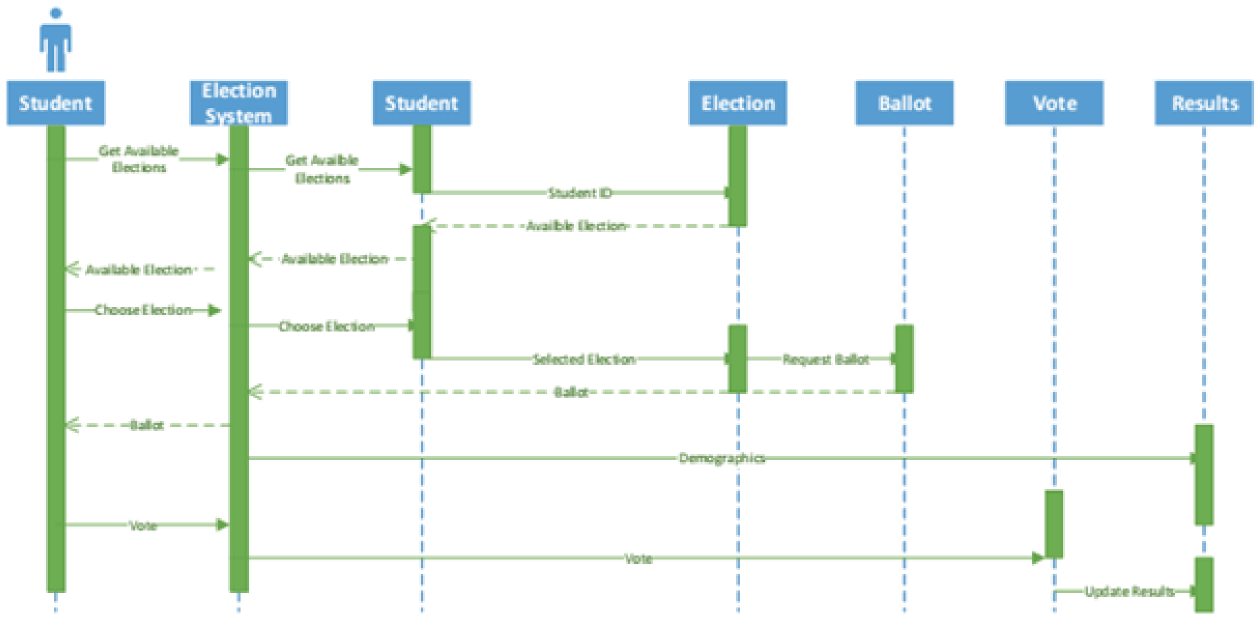
Sequence Diagram 5: Create & Modify Ballot

Figure 3: Sequence Diagram 2



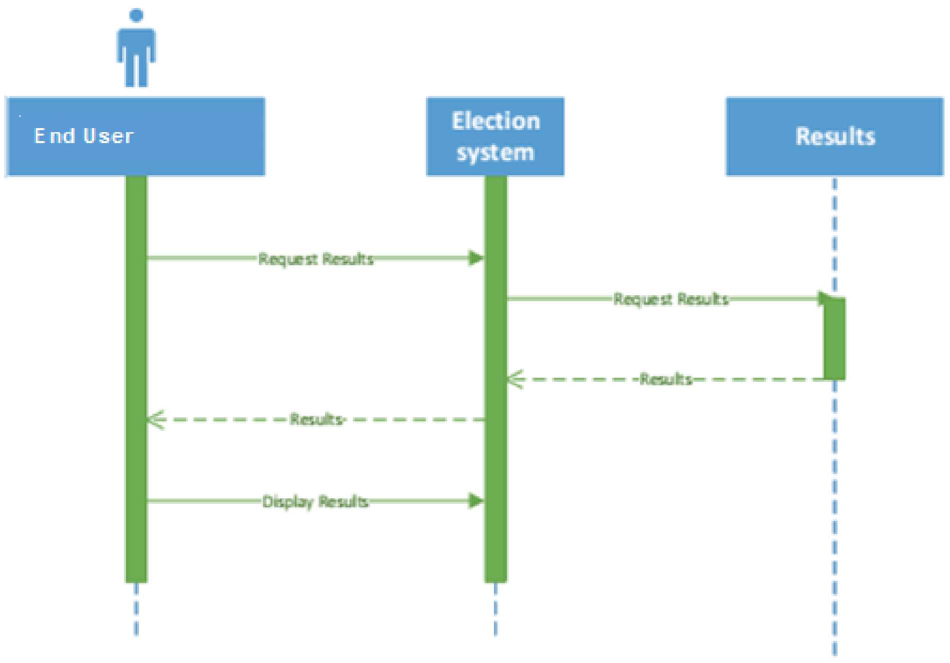
Sequence Diagram 6: Open/Close Election

Figure 4: Sequence Diagram 3



Sequence Diagram 7: Vote

Figure 5: Sequence Diagram: Vote



Sequence Diagram 8: Display Results

Figure 6: Sequence Diagram: Display Results

2.4 Activity Diagrams

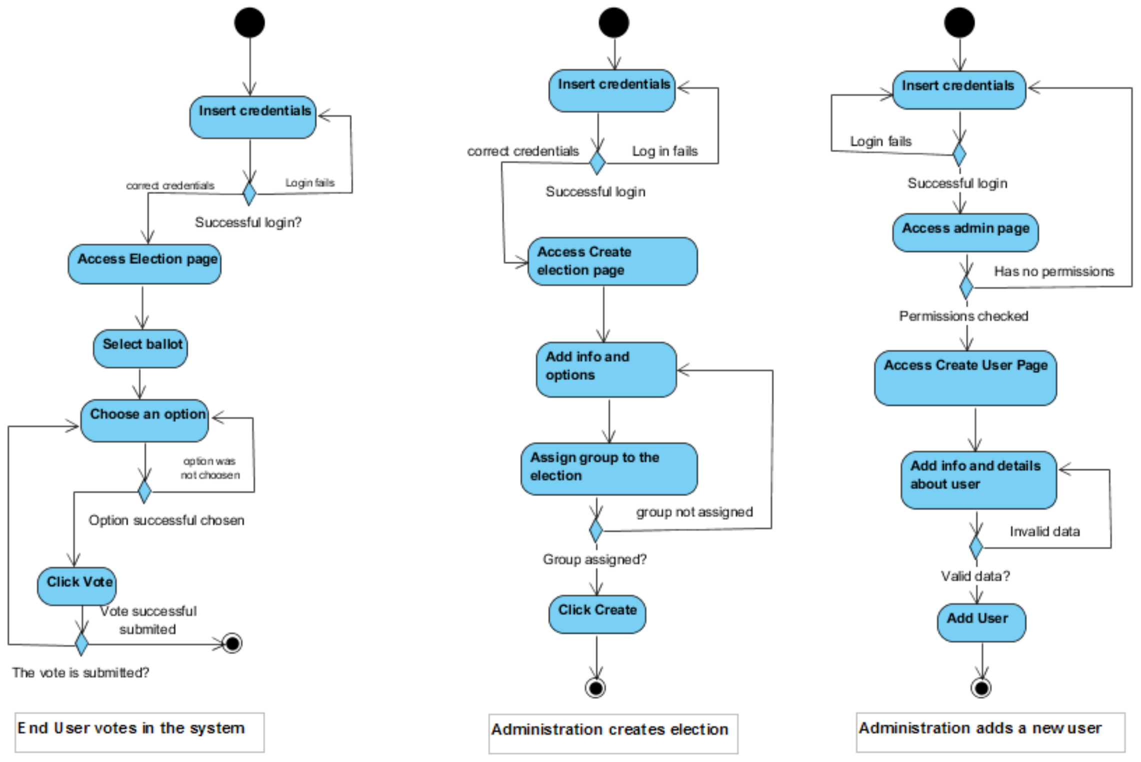


Figure 7: Activity Diagrams: End User and Administration

2.5 Components Diagram

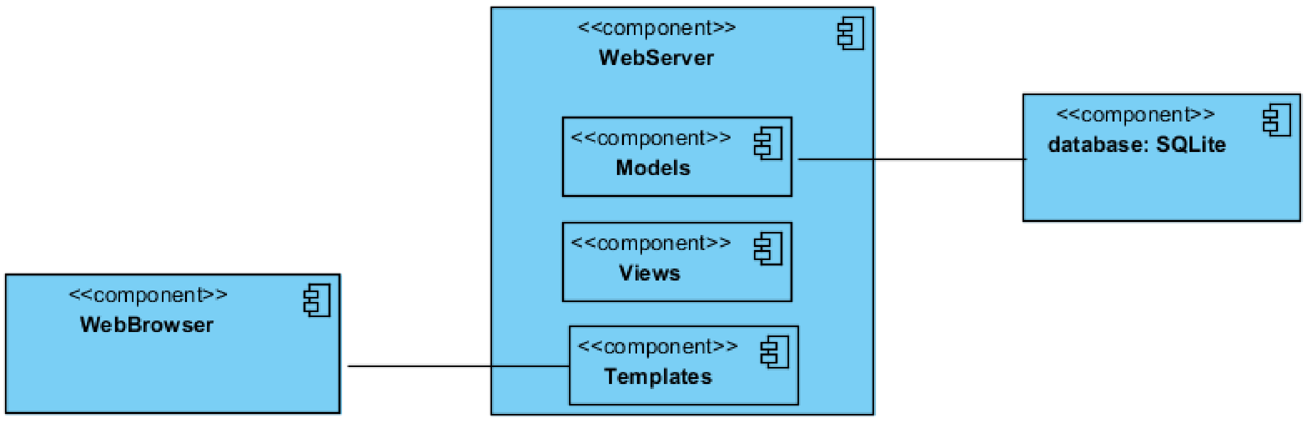


Figure 8: Component diagram: high level(browser, server and DB)

3 System Design

3.1 Packages

In the next diagram I illustrated the main packages in my system:

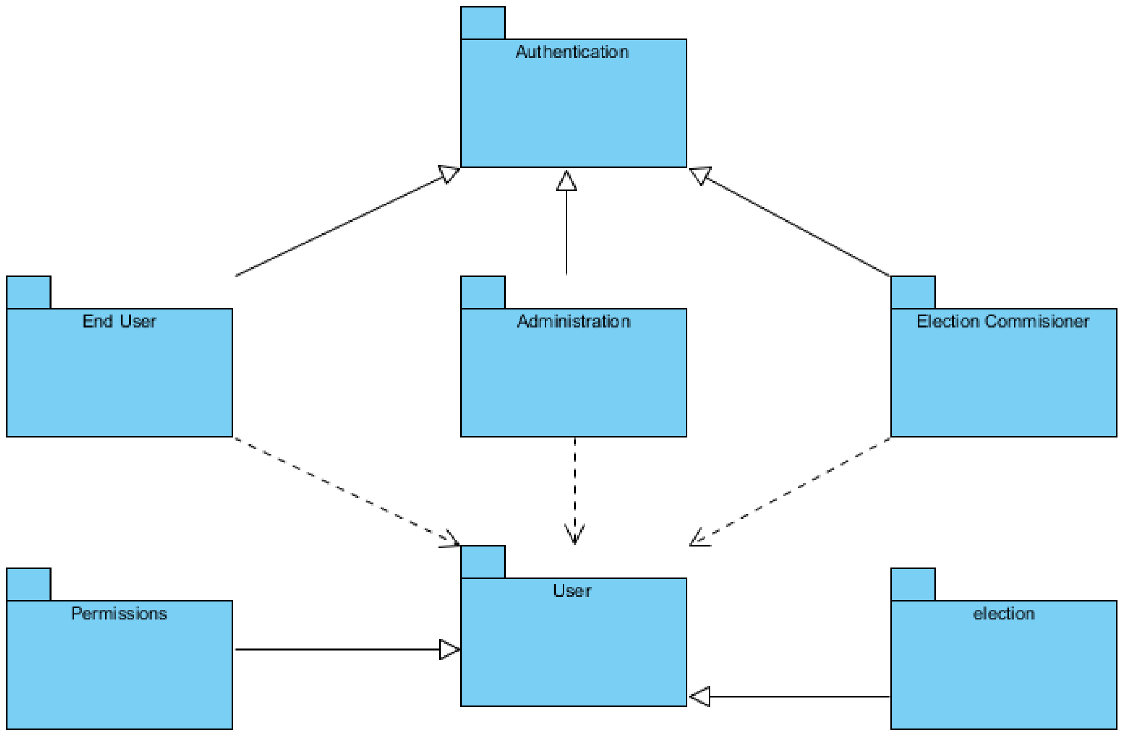


Figure 9: Main packages of the system

3.2 Deployment Diagram

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process.

3.3 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

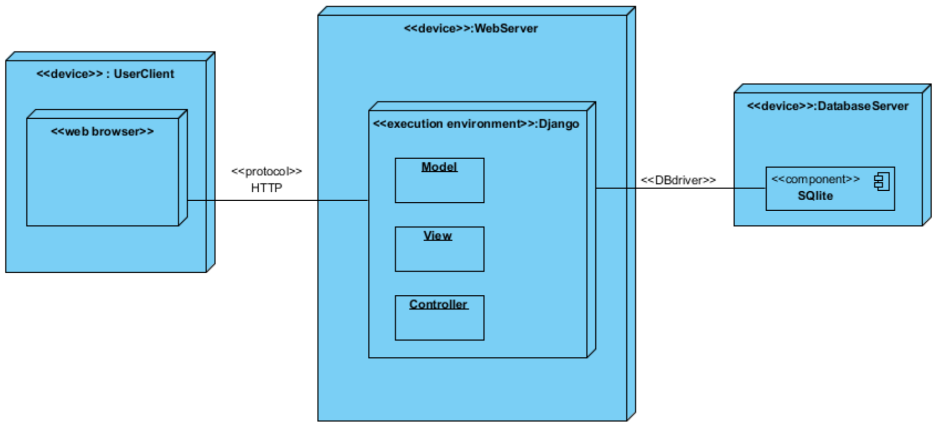


Figure 10: Deployment Diagram of the voting system

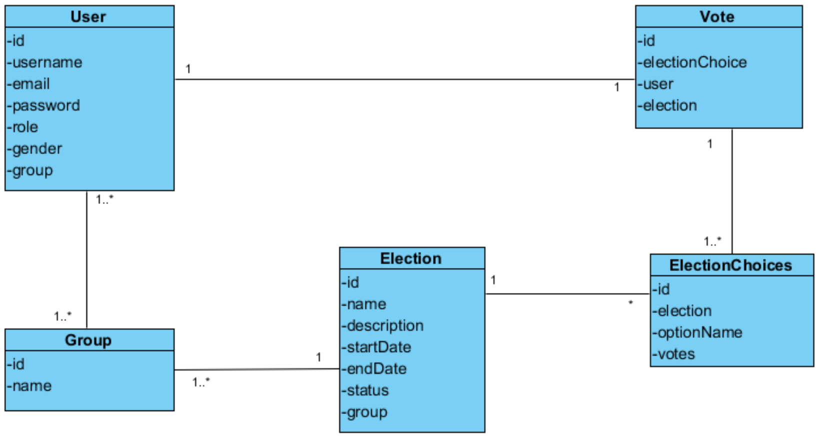
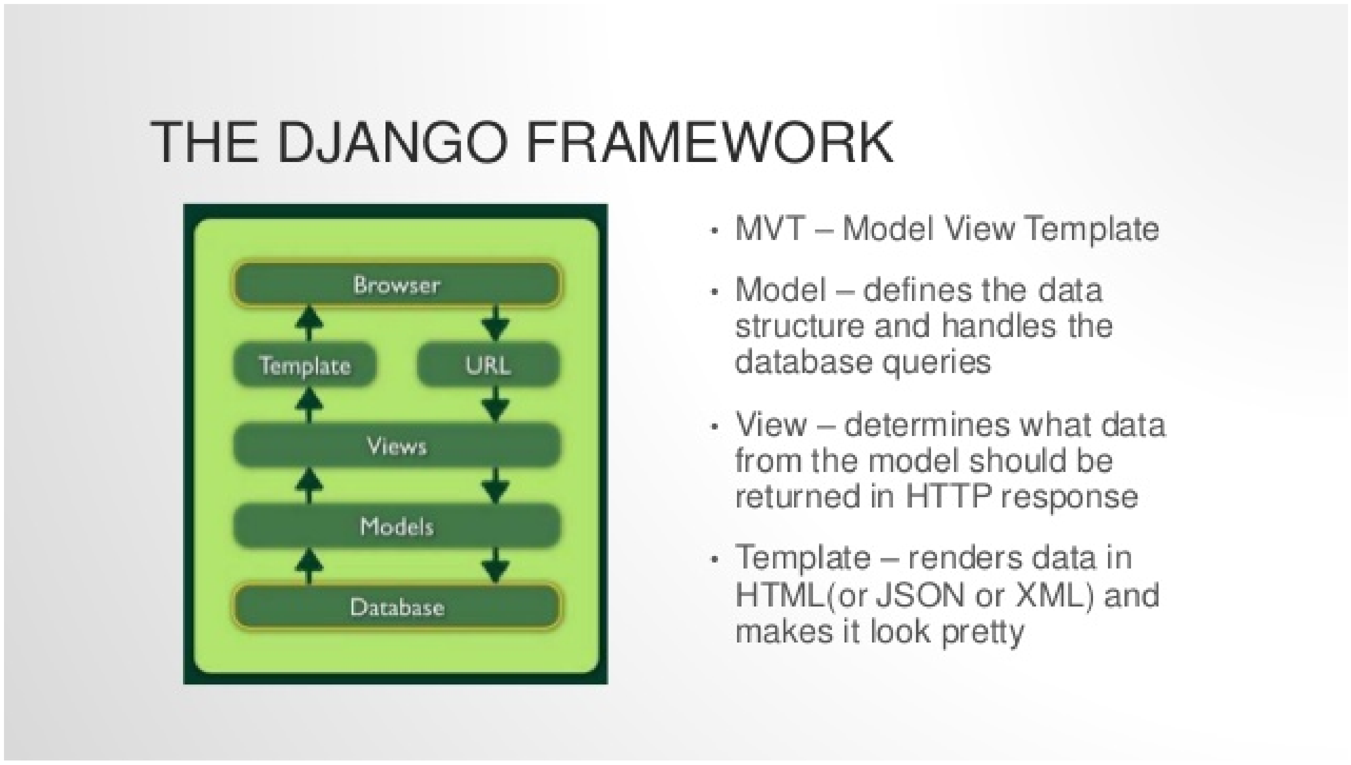


Figure 11: Class Diagram of the voting system

4 Implementation

In order to build up the system I used a python framework: Django

The following image explains perfectly how the framework works.



THE DJANGO FRAMEWORK

- MVT – Model View Template
- Model – defines the data structure and handles the database queries
- View – determines what data from the model should be returned in HTTP response
- Template – renders data in HTML(or JSON or XML) and makes it look pretty

Browser
Template
URL
Views
Models
Database

Figure 12: The Django Framework

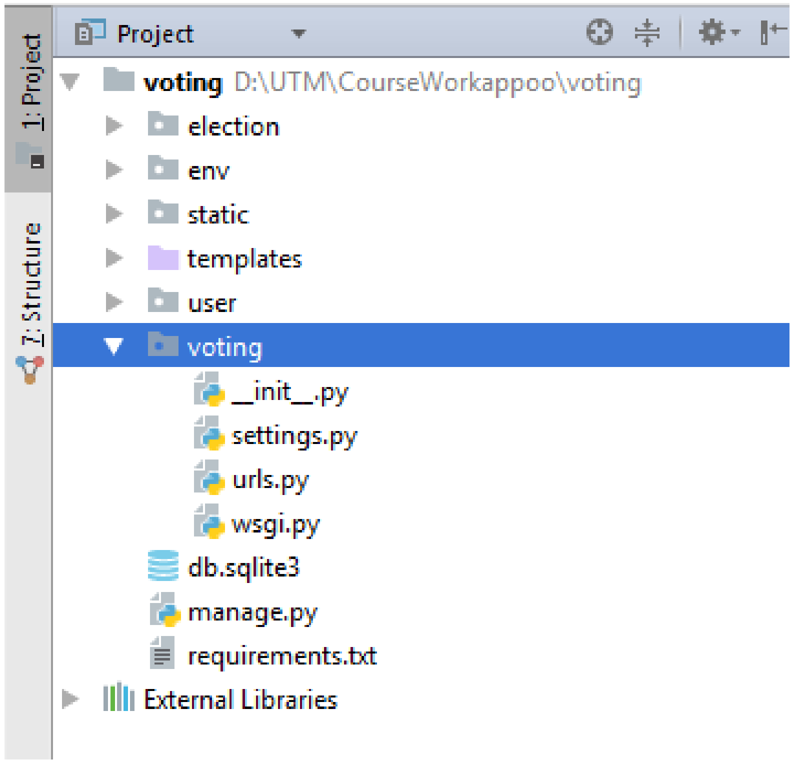
The structure of project is shown in the next image:

4.1 Defining the models

I have the following classes in my election app:

```
class Election(models.Model):


    STATUS_CHOICES =(
        (0, 'Open'),
        (1, 'Close'),
        (2, 'Canceled'),
        (3, 'Disqualified ')
```



```
voting D:\UTM\CourseWorkappoo\voting
    election
    env
    static
    templates
    user
    voting
        __init__.py
        settings.py
        urls.py
        wsgi.py
    db.sqlite3
    manage.py
    requirements.txt
External Libraries
```

Figure 13: Structure of the project

```
    )
    name = models.CharField(max_length=255) #default 5
    description = models.TextField(max_length=512) #default 1?

    startDate = models.DateTimeField()
    endDate = models.DateTimeField()
    status = models.IntegerField(choices=STATUS_CHOICES)
    group = models.ForeignKey(Group)
    #include the ballot and the election choices

class ElectionChoices(models.Model):

    election = models.ForeignKey(Election) # specify model
    optionName = models.CharField(max_length=255)
    votes = models.IntegerField(default=0) #used to count the votes for a certain
        option
```

```
class Vote(models.Model):
   user = models.ForeignKey(User) # - do we really need that?
   electionChoice = models.ForeignKey(ElectionChoices)
   election = models.ForeignKey(Election)
```

And then in my user app, I have another two classes defined:

```
class Group(models.Model):

    name = models.CharField(max_length=255)

    def __str__(self):
        return '%s - %s' % (self.id, self.name)


class User(AbstractUser):

    GENDER_CHOICES = (
        (0, ('Male')),
        (1, ('Female')),
    )

    ROLE_CHOICES = (
        (0, ('Student')),
        (1, ('Proffesor')),
        (2, ('Administration')),
        (3, ('ElectionComisioner'))

    )

    gender = models.IntegerField(choices=GENDER_CHOICES, default=0)
    role = models.IntegerField(choices=ROLE_CHOICES, default=2)
    groups = models.ManyToManyField(Group)
```

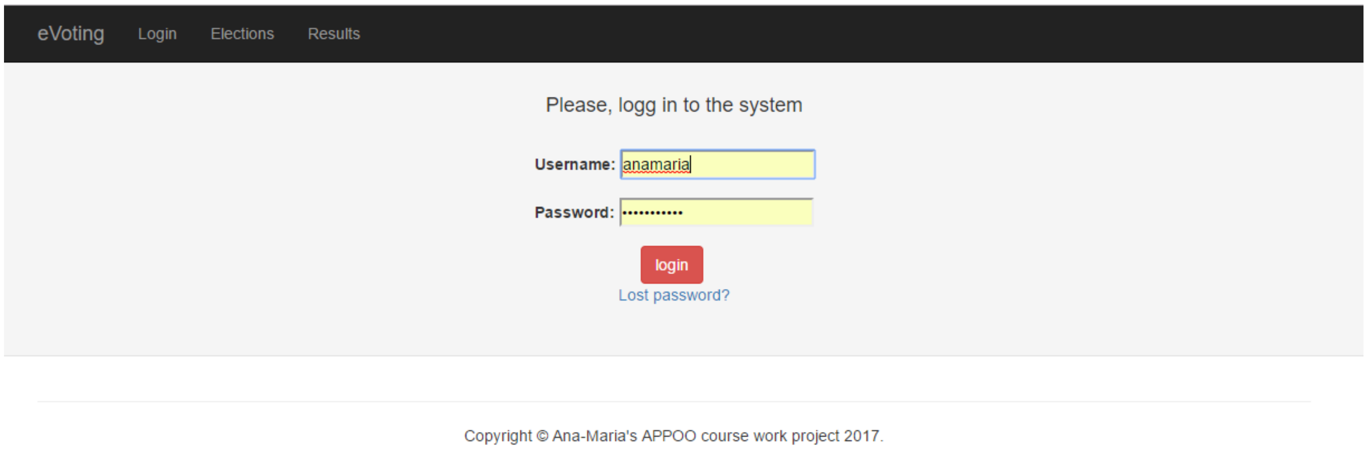
4.1.1 Views and some pages

In my **urls.py** I have the following:

```
urlpatterns = [
   url(r'^election/$', ElectionList.as_view(), name='election-name'),
   url(r'^create-election/$', CreateElection.as_view(), name='create-election'),
   url(r'^home/', MainPage.as_view(), name='main_page'), #Home page for loged in
       student
   url(r'^election/(?P<pk>\d+)$', ElectionDetail.as_view(), name='election-detail'),
   url(r'^success/$', ThanksPage.as_view(), name='success-vote'),
   url(r'^results/', Results.as_view(), name='results')
```

The system itself could be run at local host. The source code of the system can be found on github (the link is included in references).

Let's take a quick view how it looks like:



Please, logg in to the system

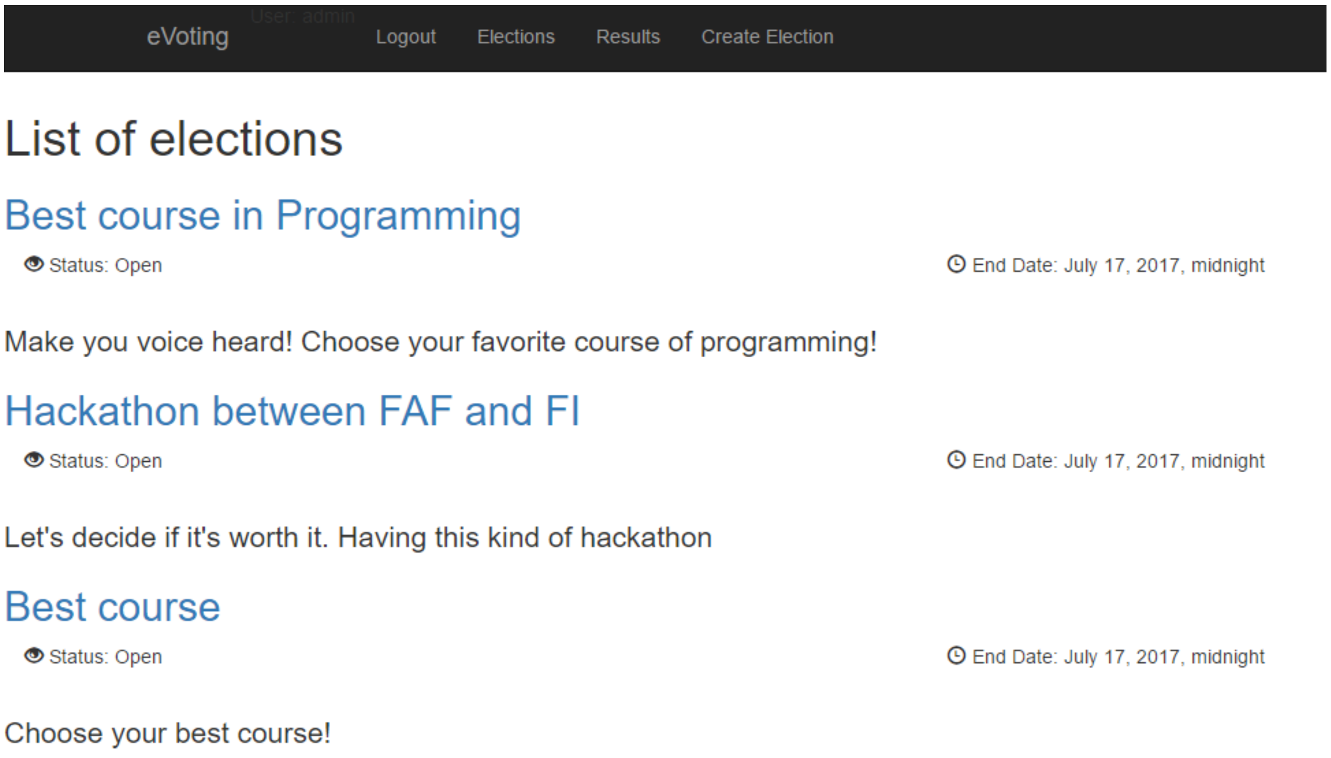
Username: anamaria

Password: •••••••••••

Copyright © Ana-Maria's APPOO course work project 2017.

Figure 14: Log into the system

It is important to notice that when creating a election the admin will be redirected to django admin page, as shown in the next image.



List of elections

Best course in Programming

Status: Open

End Date: July 17, 2017, midnight

Make you voice heard! Choose your favorite course of programming!

Hackathon between FAF and FI

Status: Open

End Date: July 17, 2017, midnight

Let's decide if it's worth it. Having this kind of hackathon

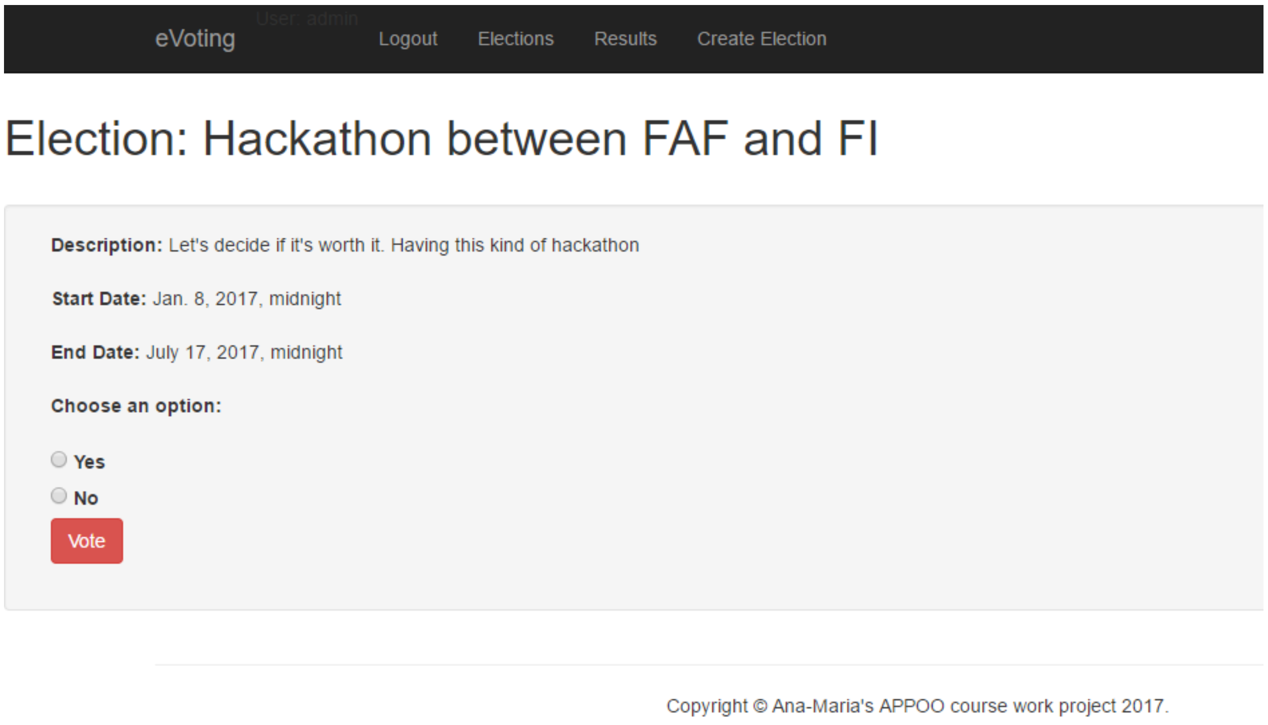
Best course

Status: Open

End Date: July 17, 2017, midnight

Choose your best course!

Figure 15: List of elections



Election: Hackathon between FAF and FI

Description: Let's decide if it's worth it. Having this kind of hackathon

Start Date: Jan. 8, 2017, midnight

End Date: July 17, 2017, midnight

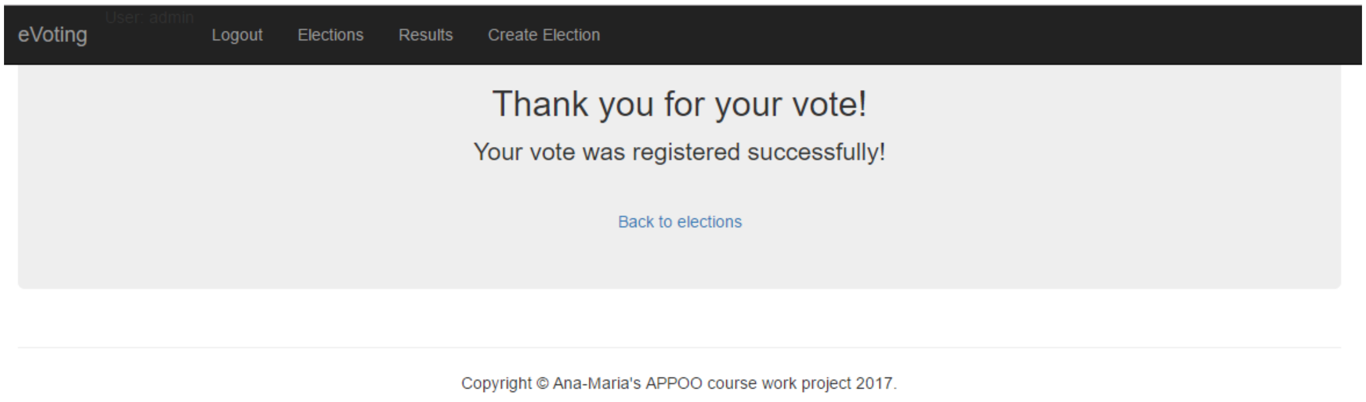
Choose an option:

Yes

No

Copyright © Ana-Maria's APPOO course work project 2017.

Figure 16: Voting Page

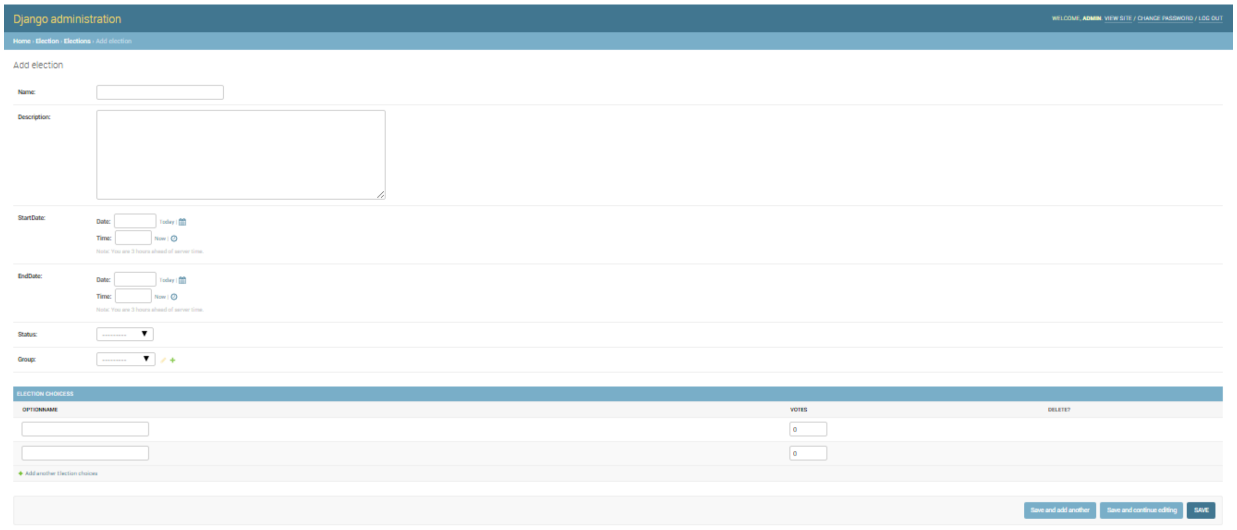


Thank you for your vote!

Your vote was registered successfully!

Copyright © Ana-Maria's APPOO course work project 2017.

Figure 17: Successful voting page



Note: You are 3 hours ahead of server time.

Note: You are 3 hours ahead of server time.

Figure 18: Add/edit election

5 Conclusion

In this work I realized how important is the modeling and design phase in the development of a system. Having an UML your work can be segnificantly reduced. UML is a modeling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on modeling OO software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects.

If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects.

Hence, the relation between OO design and UML is very important to understand. The OO design is transformed into UML diagrams according to the requirement. Before understanding the UML in detail, the OO concept should be learned properly. Once the OO analysis and design is done, the next step is very easy. The input from OO analysis and design is the input to UML diagrams.

The design and modeling phase helped me in building and implement the system. I begun with the class diagram and designing the database of the system. First, I created two separated apps **users** and **election** where I included all the models and views related to user or election. In the user app I have two main django models: **Users** and **Group**. In **election** we have 3 main classes **Election** - wich defines an election and can have more than one choice, **ElectionChoices** wich can contain more votes and respectively **Vote** which is meant to calculate the votes for each Election.

After building the models I created the Views for each model. Then the forms and the templates.

What I found really pleasand and usefull about the django was the django admin which facilitated the interaction with the system, creating and editing users, groups. Also, with some changes in **admin.py** I was able to add, change or delete elections from Django admin.

In order to build app interface I used bootstrap, which actually facilitates the process providing existing CSS and JS files.

I used PyCharm for development as an IDE and it eased my work a lot. What I liked the most is the interface, and that it made me able to commit and push my project to git (github) with just one click.

6 References

1. https://docs.microsoft.com/en-us/sql/relational-databases/views/create-indexed-views
2. https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql
3. http://docs.oracle.com/cd/E11882_01/server.112
4. http://www.w3resource.com/sql/update-views/sql-update-views.php
5. https://docs.microsoft.com/en-us/sql/relational-databases/views/create-indexed-views
6. https://www.youtube.com/channel/UCCTVrRB5KpIiK6V2GGVsR1Q