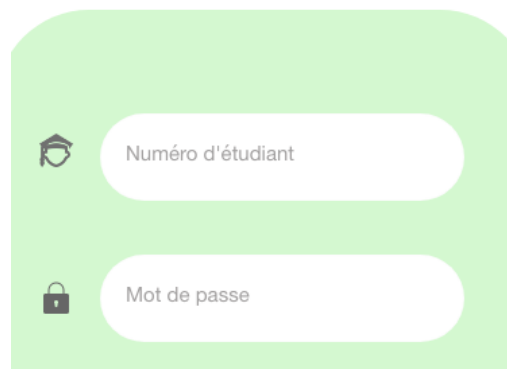


# Rapport Final

## Agenda Étudiant Jardinage



Numéro d'étudiant

Mot de passe

Réalisé par  
Hongyu YAN, Liuyi CHEN

Sous la direction de  
Mehdi CHAKHCHOUKH

Année universitaire 2020-2021  
Licence Informatique

# Introduction

En premier lieu, l'architecture du développement est basée sur le modèle Modèle Vue Contrôleur (MVC). Ce pattern permet de bien organiser le code source. Et puis nous avons utilisé scene builder et javafx pour créer les interfaces graphiques. Par ailleurs, nous avons implémenté CalendarFx pour réaliser les fonctionnalités liées au calendrier. Afin de ne pas perdre les données saisies dans l'application, nous avons stocké nos données dans des fichiers Excel.

## Fonctionnalités de l'application (Grille)

|   |   |
|---|---|
| Un affichage d'agenda basique                         | ✓ |
| Consulter, ajouter et éditer des événements           | ✓ |
| Catégorie des événements                              | ✓ |
| Évènements ponctuels et des évènements récurrents     | ✓ |
| Recherche des événements dans l'agenda                | ✓ |
| Une liste de plantes                                  | ✓ |
| Ajout, suppression et modification d'une plante       | ✓ |
| Consultation des informations détaillées d'une plante | ✓ |
| Planifier des événements spécifiques à une plante     | ✓ |
| Des informations de suivi                             | ✓ |
| Des mesures concernant chaque plante                  | ✓ |
| Affichage pour les informations météo                 | ✗ |
| Un moyen d'importer des évènements sur Google Agenda  | ✗ |

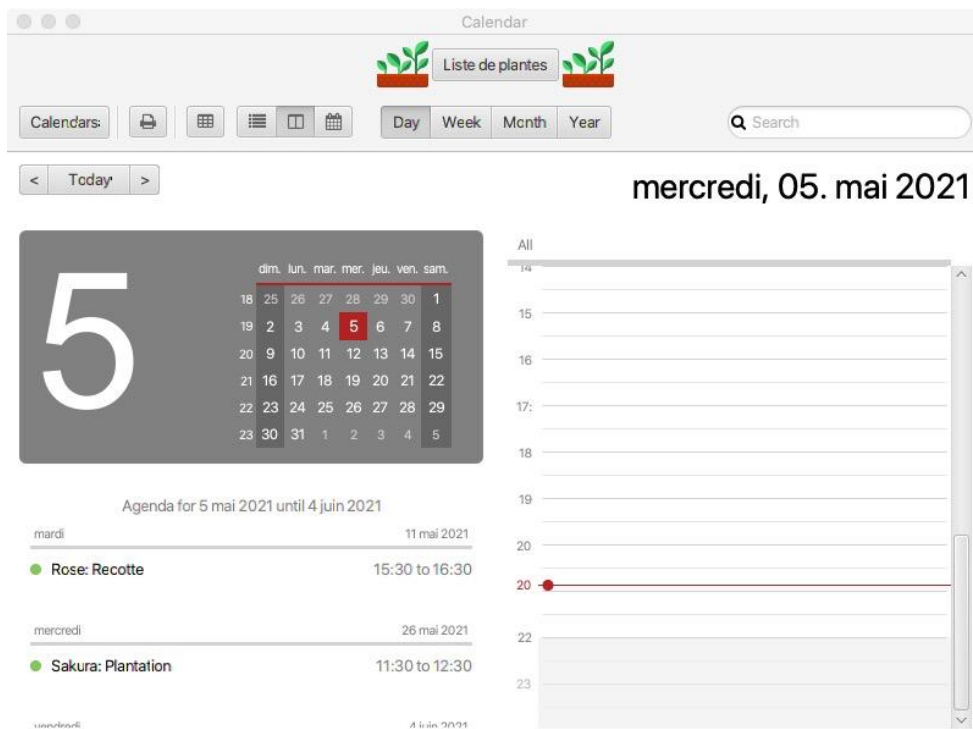
# Développement détaillé

- Calendar

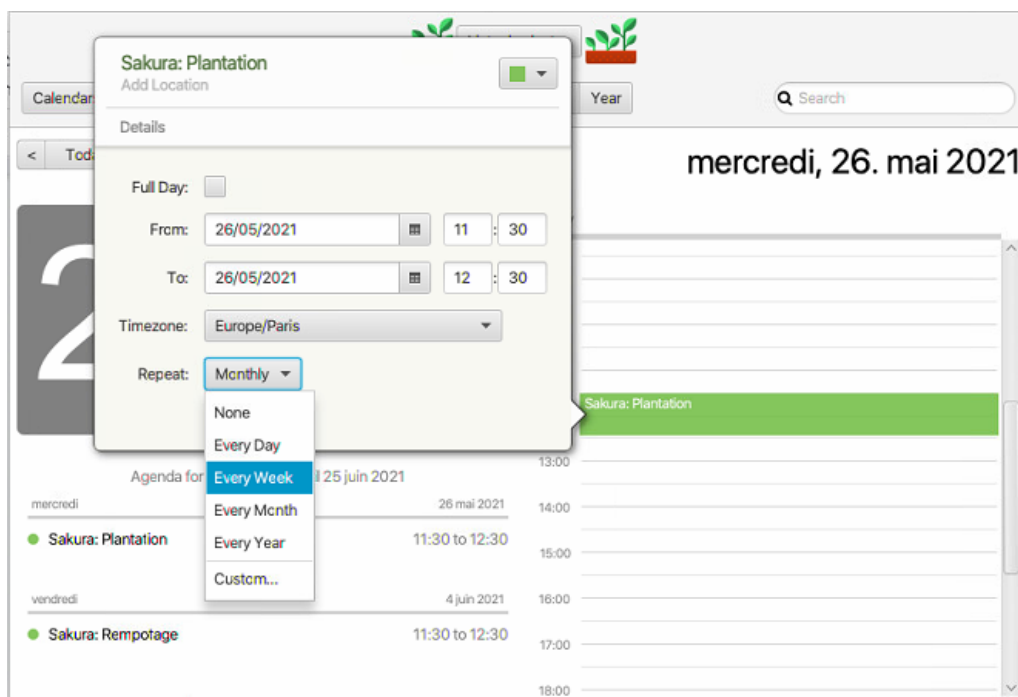
On profite du CalendarFX pour réaliser les fonctionnalités de l'agenda.

- 1) Un affichage d'agenda basique.

Quand on lance l'application, il affiche un agenda basique. Ainsi, nous pouvons accéder à la liste des plantes par le logo en haut de la page calendrier.



- 2) L'utilisateur doit pouvoir ajouter, éditer et supprimer des événements ponctuels et des événements récurrents.



### 3) Catégorie des événements.

Nous avons deux types d'événements, le bleu est le calendrier scolaire et le vert est le calendrier des plantes.

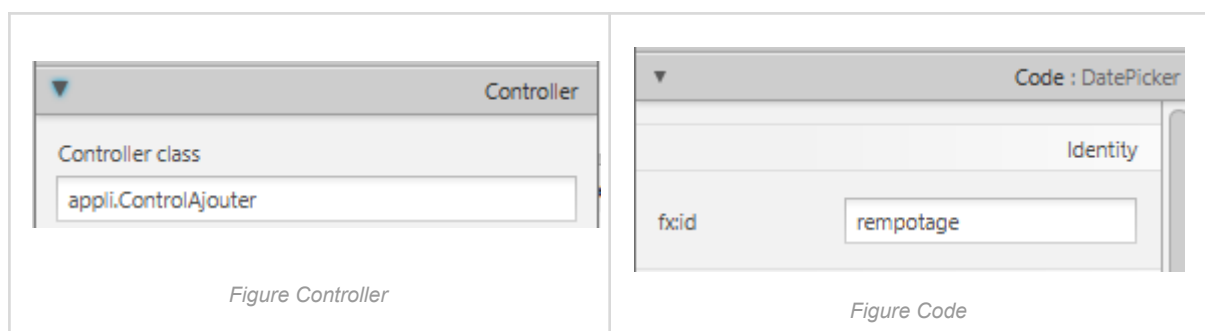
|         | dim., 9 | lun., 10 | mar., 11      | mer., 12 | jeu., 13 | ven., 14 | sam., 15 |
|---------|---------|----------|---------------|----------|----------|----------|----------|
| All Day |         |          |               |          |          |          |          |
| 13:00   |         |          |               |          |          |          |          |
| 14:00   |         |          |               |          |          |          |          |
| 15:00   |         |          |               |          |          |          |          |
| 16:00   |         |          | Rose: Recotte | anglais  |          |          |          |
| 17:00   |         |          |               |          |          |          |          |
| 18:00   |         |          | pia           |          |          |          |          |
| 19:00   |         |          |               |          |          |          |          |
| 20:00   |         |          |               |          |          |          |          |
| 21:00   |         |          |               |          |          |          |          |

### 4) Recherche des événements dans l'agenda.

#### ● Plante

Nous utilisons JavaFX Scene Builder pour concevoir et développer notre interface des plantes. Il est un outil qui permet aux utilisateurs de concevoir rapidement l'interface des applications JavaFX sans codage. Les utilisateurs peuvent faire glisser et déposer des composants dans l'espace de travail (Hierarchy), modifier leurs propriétés (Properties), appliquer des feuilles de style (Layout) et le code FXML de la mise en page sera automatiquement généré. Son résultat est un fichier FXML.

On crée des classes controller pour changer les scènes, passer les informations entre les scènes et uploader les données. Nous relient les classes java et les interfaces de Scene Builder dans la partie Controller. De plus, pour chaque composant, nous pouvons lui donner fx:id et les actions lui correspondant.



### 1) Une liste de plantes accessible facilement par l'utilisateur.

Lorsque nous cliquons sur le bouton(en forme d'une plante) en haut de l'interface du calendrier, nous allons dans une page contenant toutes les plantes. De même, si on clique sur le bouton calendrier sur cette page, nous pouvons revenir à notre calendrier.

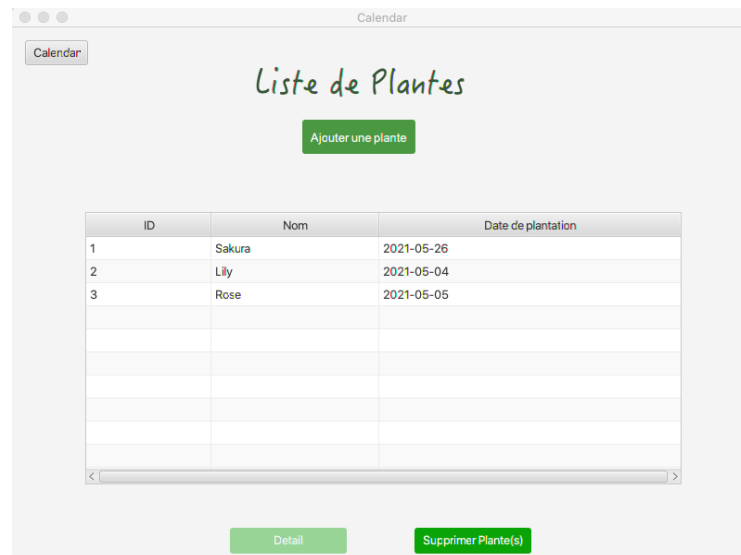


Figure Liste de plante

## 2) Consultation les informations détaillées d'une plante de la liste.

Si l'utilisateur a besoin de visualiser les informations spécifiques d'une certaine plante, il suffit de sélectionner cette plante dans le tableau, puis cliquez sur le bouton de "détail". L'utilisateur passera à la page "Détail" de l'installation.

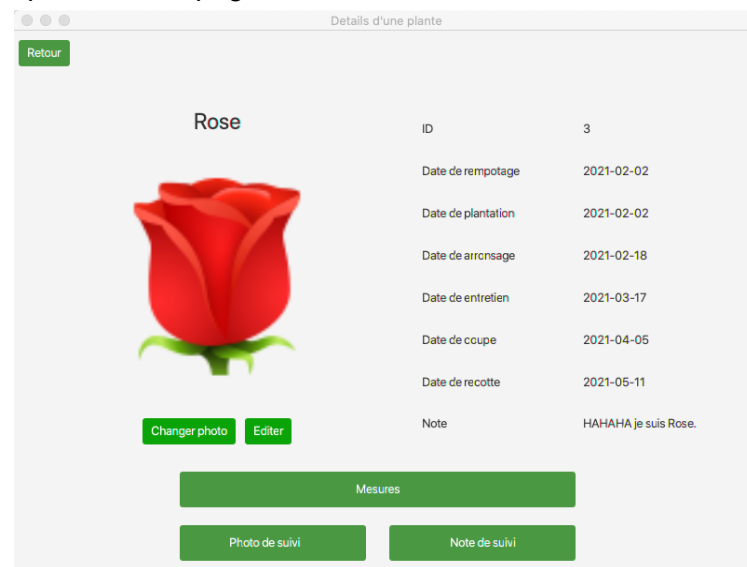


Figure Détail de plante

## 3) Ajout et suppression d'une plante.

Il y a deux boutons sur la page "Liste de plantes" qui permettent d'ajouter et de supprimer des plantes respectivement. Si on veut ajouter une plante, on a juste besoin de cliquer sur "Ajouter une plante". La suppression ressemble à la consultation des informations. Si vous souhaitez supprimer une plante, il suffit de la sélectionner dans la table, puis de cliquer sur "Supprimer Plante(s)".

Ajout d'une plante

Retour

Nom

Photo [Choisir une photo](#)

Dates des événements Notes

Date de repotage

Date de plantation

Date de arrosage

Date de entretien

Date de coupe

Date de recotte

enregistrer

Figure Liste de plante

#### 4) Modification des informations d'une plante

Dans la page de détail, nous avons un bouton "Edit" qui nous tourne à la page d'édit. Cette page nous permet de modifier les photos, les notes et les dates des événements. Nous pouvons également mettre à jour les mesures en cliquant les éléments de la liste de la page "Mesure".

Modifier une plante

Retour

Nom

Photo [Choisir une photo](#)

Dates des événements Notes Photo

Date de repotage

Date de plantation

Date de arrosage

Date de entretien

Date de coupe

Date de recotte

Mesures

enregistrer

Figure Liste de plante

Page Mesures

Retour

Sakura / Measure

Search

| Id | Nom     | Valeur | Unite | Date       |
|----|---------|--------|-------|------------|
| 1  | Hauteur | 12     | cm    | 2021-03-04 |
|    |         |        |       |            |
|    |         |        |       |            |
|    |         |        |       |            |
|    |         |        |       |            |
|    |         |        |       |            |
|    |         |        |       |            |
|    |         |        |       |            |

Hauteur

Ajouter nouvelle mesure

Supprimer mesure(s)

Figure Mesures

##### 5) Planification des événements spécifiques.

Comme ce qui est présent au-dessous, nous pouvons définir les dates de plantation, rempotage, arrosage, entretien, coupe, récolte en ajoutant ou modifiant les informations d'une plante.

##### 6) Consultation des informations de suivi

On a les boutons "Photo de suivi" et "Note de suivi" permettant d'aller aux pages de suivi pour consulter les photos et les notes concernant cette plante. Lorsqu'on met à jour la photo ou la note actuelle de la plante, on ajoute les nouvelles informations dans les pages de suivi.

Photo de suivi

Retour

Photo

##### 7) Sauvegarde des données dans le fichier

# Rapport technique

- Création des events dans le calendrier

```
public void createEvent(Calendar jardins, LocalDate date, String str) {
    Entry<String> entry = new Entry<>(str);
    entry.setInterval(LocalDate.now());
    entry.changeStartDate(date);
    entry.changeEndDate(date);
    Random rand = new Random();
    int randomNum = rand.nextInt(8) + 9;
    entry.changeStartTime(LocalTime.of(randomNum, 30));
    entry.changeEndTime(LocalTime.of(randomNum+1, 30));
    jardins.addEntry(entry);
}
```

- Basculer entre différentes interfaces

Nous avons utilisé FXMLLoader pour changer la page comme la suivante.

```
void detailPlante(ActionEvent event) throws IOException {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(getClass().getResource("view/DetailPlante.fxml"));
    Parent detailPlante = loader.load();

    // access the controller and call a method
    ControlDetail controller = loader.getController();
    controller.initData(tableView.getSelectionModel().getSelectedItem());

    Stage window = (Stage)detailBut.getScene().getWindow();
    window.setTitle("Details d'une plante");
    window.setScene(new Scene(detailPlante));
    window.centerOnScreen();
    window.show();
}
```

- Affichage en implémentant l'interface "Initializable"

Grâce à la méthode "initialize" de l'interface "Initializable", notre interface graphique puis être initialisé.

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    try {
        getPlante();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
    nomColumn.setCellValueFactory(new PropertyValueFactory<Plante, String>("Nom"));
    idColumn.setCellValueFactory(new PropertyValueFactory<Plante, String>("id"));
    plantationColumn.setCellValueFactory(new PropertyValueFactory<Plante, LocalDate>("plantation"));

    tableView.setItems(plante);

    // Update the table to allow for the nom field
    tableView.setEditable(true);
    nomColumn.setCellFactory(TextFieldTableCell.forTableColumn());
    idColumn.setCellFactory(TextFieldTableCell.forTableColumn());

    // This will allow the table to select multiple rows at once
    tableView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);

    // Disable the detailBut until a row is selected
    this.detailBut.setDisable(true);
}
```

- Choisir les photos depuis l'ordinateur

```
public void choisirPhoto(MouseEvent event) {
    Stage stage = new Stage();
    filechooser = new FileChooser();
    filechooser.setTitle("Open image");
    this.filePath = filechooser.showOpenDialog(stage);
    if (filePath != null) {
        path = filePath.toURI().toString();
    }
}
```



- Une alerte lors de la suppression

Lorsque l'on veut supprimer un élément, l'application va afficher une alerte pour que l'utilisateur puisse confirmer/annuler la suppression.

```
public boolean supWarning() {
    Alert supp = new Alert(AlertType.CONFIRMATION);
    supp.setTitle("Suppression");
    supp.setHeaderText("êtes-vous sûr de vouloir supprimer ?");
    Optional<ButtonType> result=supp.showAndWait();
    return result.get()==ButtonType.OK;
}
```

- Contrainte pour les entrée numérique

Lors de la création de la valeur de mesure, nous ne pouvons saisir que des nombres

```
//permet seulement les ints ou floats
valTextField.textProperty().addListener(new ChangeListener<String>() {
    @Override
    public void changed(ObservableValue<? extends String> observable, String oldValue,
        String newValue) {
        if (!newValue.matches("\\d+?\\.?\\d*")) {
            valTextField.setText(newValue.replaceAll("[^\\d]", ""));
        }
    }
});
```

- Recherche dans la table des mesures

```
//Filtre List
FilteredList<Measure> filteredData = new FilteredList<Measure>(mesuresList,b-> true);
searchField.setOnKeyReleased(e->{
    searchField.textProperty().addListener((observable,oldValue,newValue)->{
        filteredData.setPredicate(mesure ->{
            if (newValue == null || newValue.isEmpty()) {
                return true;
            }

            String lowerCaseFilter = newValue.toLowerCase();

            if(mesure.getNom().toLowerCase().indexOf(lowerCaseFilter)!= -1) {
                return true;
            }else if (mesure.getUnite().toLowerCase().indexOf(lowerCaseFilter) != -1) {
                return true;
            }else if(String.valueOf(mesure.getValeur()).indexOf(lowerCaseFilter)!=-1) {
                return true;
            }else
                return false;
        });
    });
tableView.setItems(filteredData);
SortedList<Measure> sortedData = new SortedList<Measure>(filteredData);
sortedData.comparatorProperty().bind(tableView.comparatorProperty());
tableView.setItems(sortedData);
});
```

- Utilisation des excels en tant que la base pour enregistrer et récupérer les données:

Lorsque l'on modifie les informations des plantes (ajouter/ éditer/ supprimer), on met à jour également les fichiers excels correspondants. Donc on ne va pas perdre les informations si on relance l'application. Par exemple, on ouvre le fichier plante pour lire des informations dedans et puis le fermer

```
File measureFile = new File("src/data/plantes.xlsx");
FileInputStream fis = new FileInputStream(measureFile);
XSSFWorkbook workbook = new XSSFWorkbook(fis);
XSSFSheet sheet = workbook.getSheetAt(0);
```

```
workbook.close();  
fis.close();  
...
```

De même pour modifier la base (par exemple, modifier la liste de plante):

```
//create a new plant  
Plante newPlante = new Plante(newId,nom.getText());  
File plante_File=new File("src/data/plantes.xlsx");  
FileInputStream fis_plante = new FileInputStream(plante_File);  
XSSFWorkbook wb_plante = new XSSFWorkbook(fis_plante);  
XSSFSheet sheet_plante = wb_plante.getSheetAt(0);  
  
FileOutputStream outputStream = new FileOutputStream("src/data/plantes.xlsx");  
wb_plante.write(outputStream);  
wb_plante.close();  
fis_plante.close();
```

## Bibliothèques

CanendarFx :

<https://dlsc.com/products/calendarfx/>

Tutoriel:

<https://www.youtube.com/watch?v=STIHzuVmIG4&list=PLoodc-fmtJNYbs-gYCdd5MYS4CKVbGHv2>