Access Keys Documentation

Login System

Login

When logging in, there are four options:

- Cashier
- Barista
- Admin
- Display

And as it is noticeable, they are the same options on the “Add key” view, and to make it function the if statement below was implanted on the Login View Controller.

```
27    @FXML private void onEnter()
28    {
29      String pwd = loginViewModel.getUserType();
30      System.out.println("ViewController: " + pwd);
31      if (pwd.equals("Barista"))
32      {
33        getViewHandler().openView( id: "BaristaView.fxml");
34      }
35      else if (pwd.equals("Admin"))
36      {...}
39      else if (pwd.equals("Cashier"))
40      {...}
43      else if (pwd.equals("Display"))
44      {...}
47      else
48      {...}
```

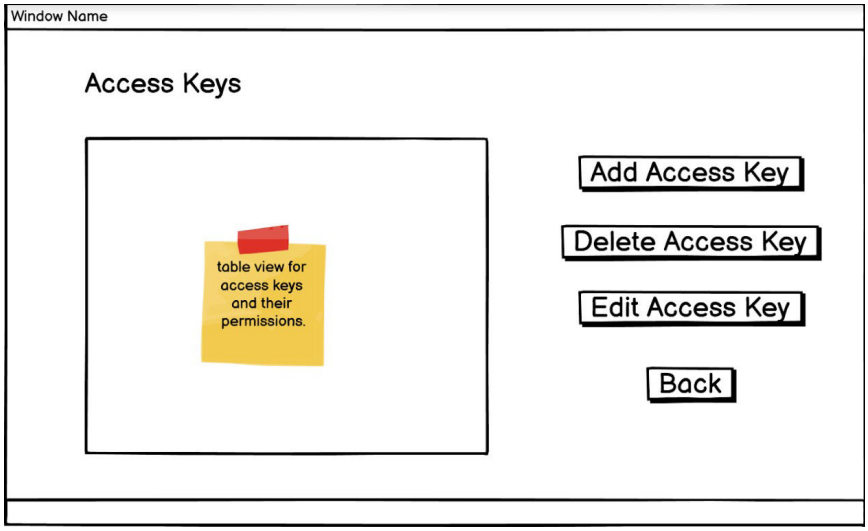
Manage Access Keys

Design Choices

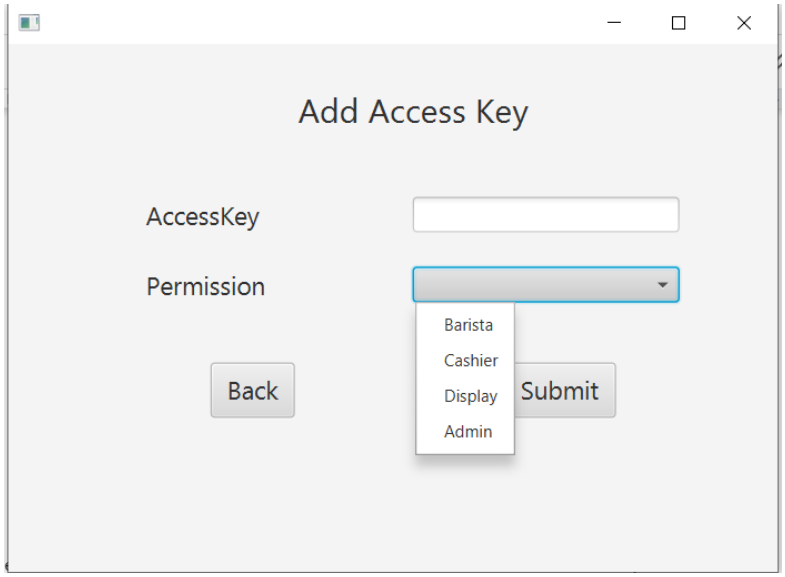
The product backlog Items that this view fulfils are the following:

- As an admin, I want to manage access keys to the system so that limited functionalities are not accessible to everyone

For the purpose of making this possible, the below window was create with a table filled with all the access keys and buttons that make the admin able to add and remove the selected access key.



To make the “add access key” possible, another view was created:



IMPLEMENTATION

Starting with the “main view”, it as a normal view controller.

The instance variables are:

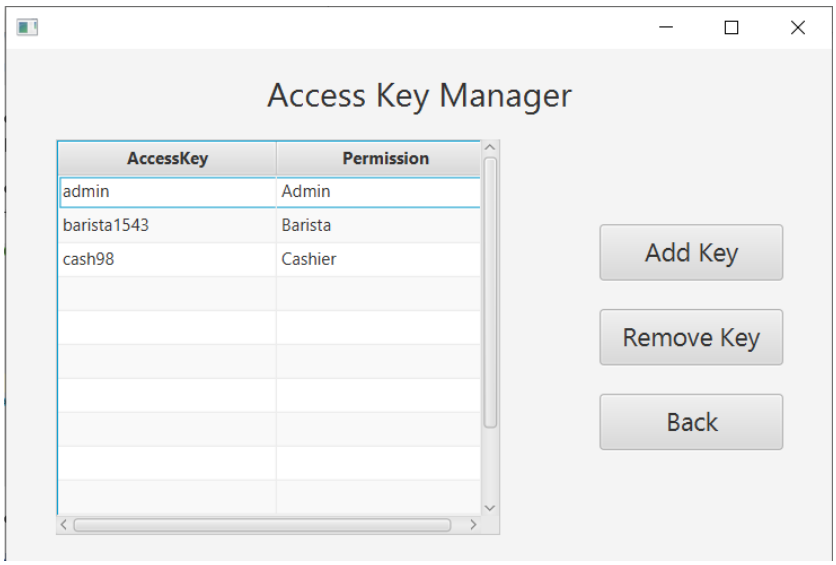
```
18  public class ManageAccessKeyViewController extends  ViewController
19  {
20
21    @FXML private TableView accessKeyTable;
22    @FXML private TableColumn accessKey;
23    @FXML private TableColumn permissionKey;
24    private ManageAccessKeysViewModel viewModel;
```

The init() method sets the Cell Factories for the TableView, calls the reset() method (screenshot below) and initializes the selectionModel.

The reset() method calls the reset() method in the viewModel, making it so that the View package is not concerned with implementation in any way.

```
26  @Override protected void init()
27  {
28  viewModel = getViewModelFactory().getManageAccessKeysViewModel();
29  accessKey.setCellValueFactory(new PropertyValueFactory<AccessKeyProperty, StringProperty>( s: "accessKey"));
30  permissionKey.setCellValueFactory(new PropertyValueFactory<AccessKeyProperty, StringProperty>( s: "permissionKey"));
31
32  reset();
33  }
34  public void reset()
35  {
36    viewModel.reset();
37    accessKeyTable.setItems(viewModel.getAllAccessKeys());
38  }
```

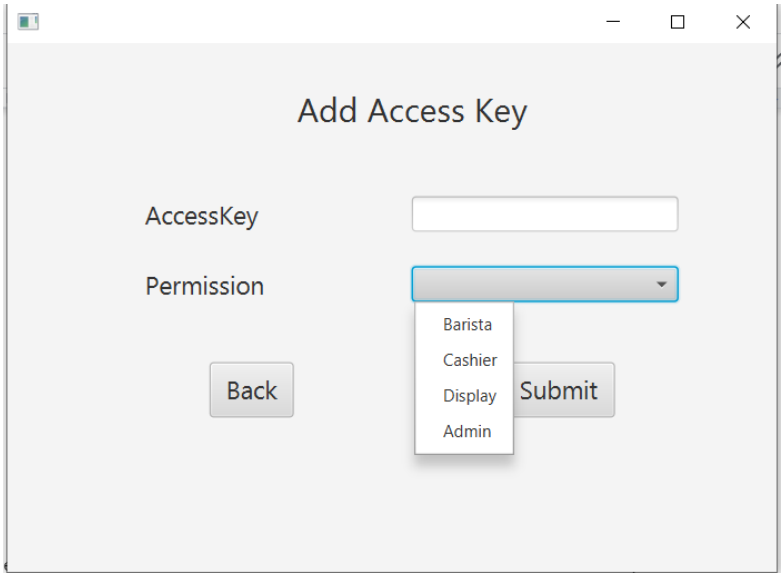
The view can be seen in the image below:



As it can be seen, the button edit was removed because of redundance

Add

Just as soon as the “Add key” is presses, the view below will open:



This view as instance variables:

```
11  public class AddAccessKeyViewController extends ViewController
12  {
13    @FXML private TextField accessKey;
14    @FXML private ChoiceBox<String> permissionChoiceBox;
15    @FXML private Label errorLabel;
16    private AddAccessKeyViewModel addAccessKeyViewModel;
```

And more important is the init(), that binds not only the textbox but also the permission checkbox so It fills with the options shown on the image above.

```
18  @Override protected void init()
19  {
20    addAccessKeyViewModel = getViewModelFactory().getAddAccessKeyViewModel();
21    accessKey.textProperty().bindBidirectional(addAccessKeyViewModel.accessKeyProperty());
22    errorLabel.textProperty()
23        .bindBidirectional(addAccessKeyViewModel.errorProperty());
24    permissionChoiceBox.getItems().addAll(addAccessKeyViewModel.getPermission());
25    //Putting items inside the choice box
26    permissionChoiceBox.valueProperty()
27        .bindBidirectional(addAccessKeyViewModel.chosenProperty());
28    //Binding the chosen item with the view model
29  }
```

Remove

When the “Remove Key” button is pressed, the selected access key is deleted from the table and the database.

Server-side and Database

In terms of passing the information from the client to the server and database , it works in the same exact way as the items and orders