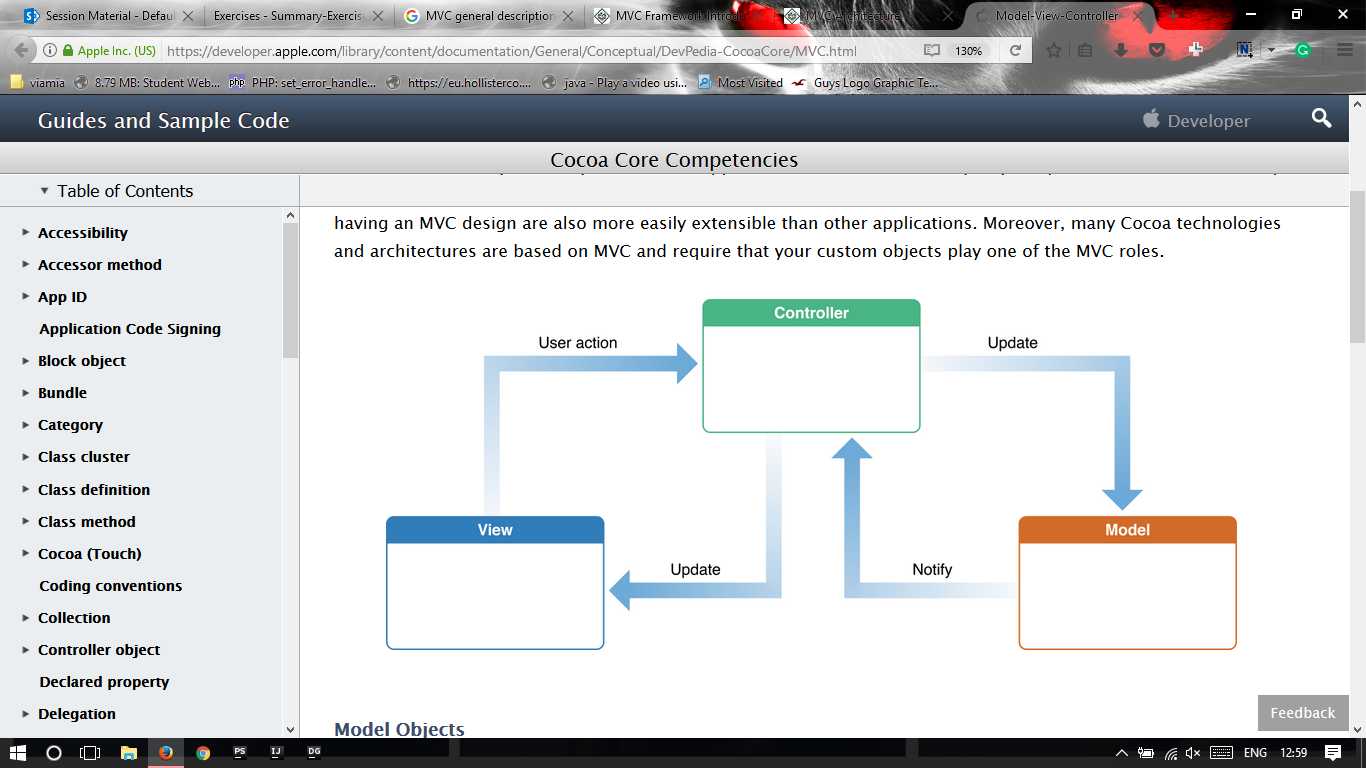
# **Model-View-Controller design pattern**

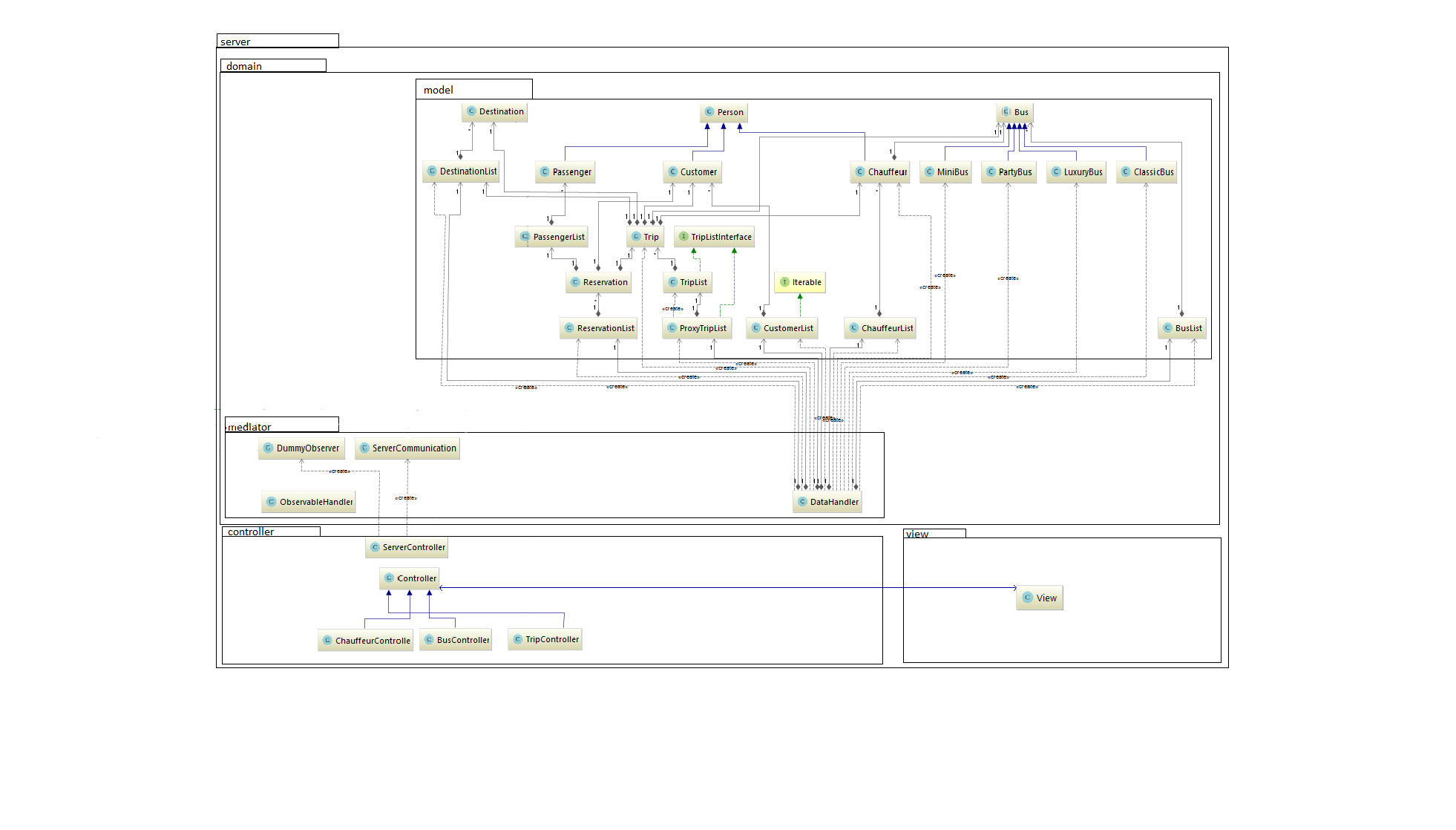
## **General diagram for MVC design pattern:**

****

## **MVC description:**

* The Model-View-Controller (MVC) design pattern assigns objects in an application one of three roles: model, view, or controller. The pattern defines not only the roles objects play in the application, it defines the way objects communicate with each other.
* **Controller**
  + Controllers act as an interface between Model and View components to process all the logic and requests, manipulate data using the Model component and interact with the Views to show the final output.
* **View**
  + Views are responsible for displaying all or a portion of the data to the user.
* **Model**
  + Model corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other logic-related data.

## **Our implementation:**



* **Controller** (BusController)

private void loadList() {  
 busListview.getSelectionModel().setSelectionMode(SelectionMode.*MULTIPLE*);  
 ObservableList<Bus> items = DataHandler.*getInstance*().getObservableListOfBuses();  
 busListview.setItems(items);  
}  
  
*/\*\*  
 \* Deletes selected buses from list of buses.  
 \*/*public void deleteBus(ActionEvent actionEvent) throws FileNotFoundException, ParseException {  
 ObservableList<Bus> selected;  
 selected = busListview.getSelectionModel().getSelectedItems();  
 for (Bus aSelected : selected) {  
 DataHandler.*getInstance*().removeFromBuslist(aSelected);  
 }  
  
 loadList();  
 DataHandler.*getInstance*().save();  
}

}  
 }

* BusController delegates work to DataHandler from mediator package in order to get all data it needs to display something in GUI, or to create new objects of classes in model package.
* DataHandler class:

public void removeFromBuslist(Bus bus) {  
 busList.removeBus(bus);  
}

public ObservableList getObservableListOfBuses() {  
 ObservableList<Bus> items = FXCollections.*observableArrayList*();  
 for (Bus bus : busList.getArrayBuses()) {  
 items.add(bus);  
 }  
 return items;  
}

* DataHandler solves the tasks by accessing the model and it contains all the necessary methods to mediate work from controller classes to model classes.
* The package View contains all fxml files for all different screens in the system. Those do not have access to anything else in the system, and are controlled by corresponding classes in the Controller package.