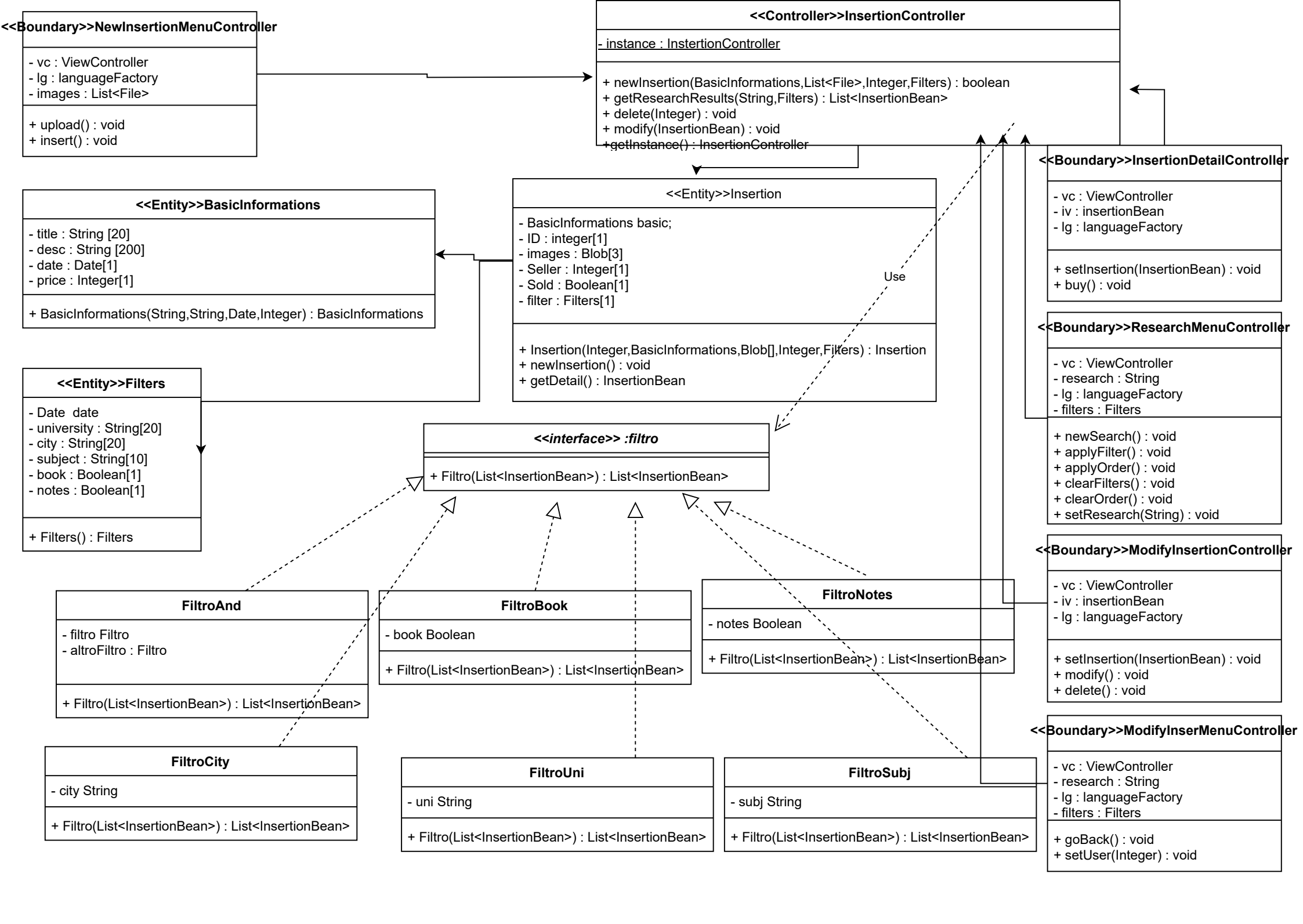
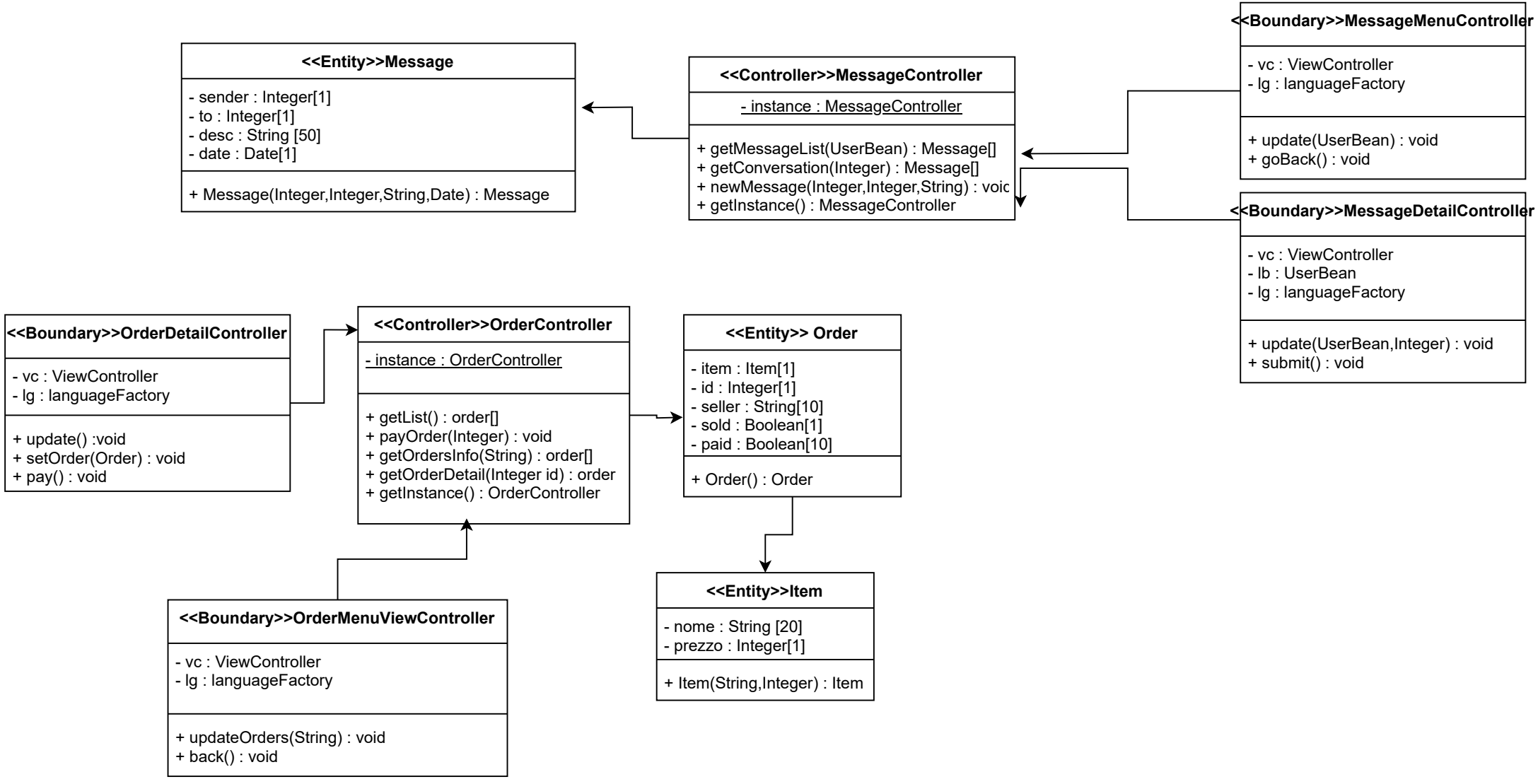
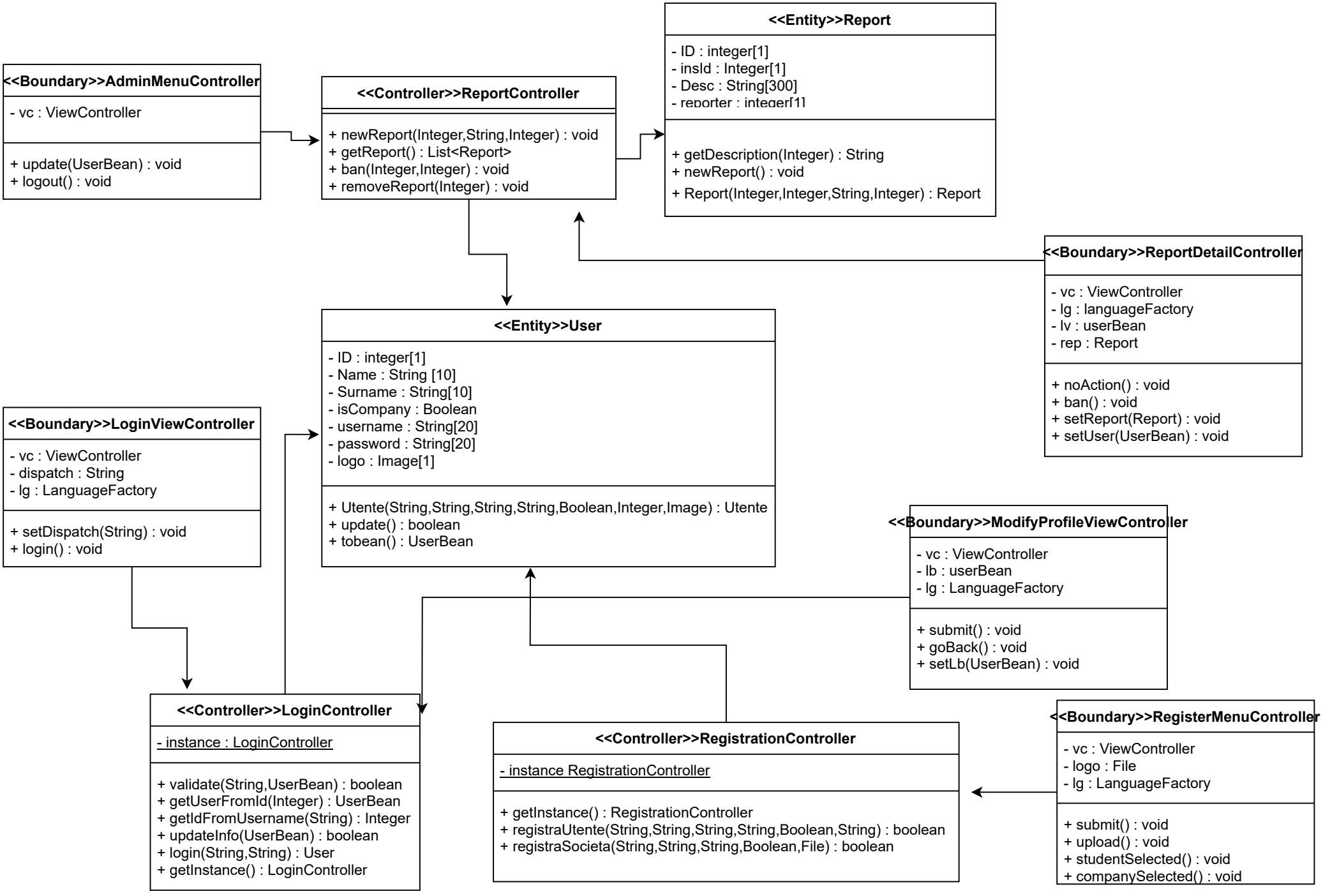
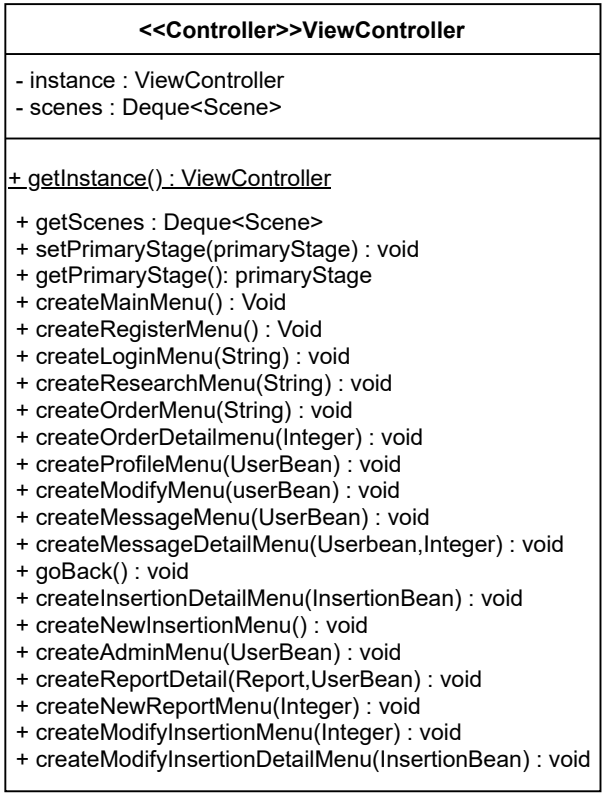

<<Boundary>>NewInsertionMenuController
- vc : ViewController
- lg : languageFactory
- images : List<File>
+ upload() : void
+ insert() : void
<<Controller>>InsertionController
- instance : InstertionController
+ newInsertion(BasicInformations,List<File>,Integer,Filters) : boolean
+ getResearchResults(String,Filters) : List<InsertionBean>
+ delete(Integer) : void
+ modify(InsertionBean) : void
+getInstance() : InsertionController
<<Boundary>>InsertionDetailController
- vc : ViewController
- iv : insertionBean
- lg : languageFactory
+ setInsertion(InsertionBean) : void
+ buy() : void
<<Entity>>BasicInformations
- title : String [20]
- desc : String [200]
- date : Date[1]
- price : Integer[1]
+ BasicInformations(String,String,Date,Integer) : BasicInformations
<<Entity>>Insertion
- BasicInformations basic;
- ID : integer[1]
- images : Blob[3]
- Seller : Integer[1]
- Sold : Boolean[1]
- filter : Filters[1]
+ Insertion(Integer,BasicInformations,Blob[],Integer,Filters) : Insertion
+ newInsertion() : void
+ getDetail() : InsertionBean
Use
<<Boundary>>ResearchMenuController
- vc : ViewController
- research : String
- lg : languageFactory
- filters : Filters
+ newSearch() : void
+ applyFilter() : void
+ applyOrder() : void
+ clearFilters() : void
+ clearOrder() : void
+ setResearch(String) : void
<<Entity>>Filters
- Date date
- university : String[20]
- city : String[20]
- subject : String[10]
- book : Boolean[1]
- notes : Boolean[1]
+ Filters() : Filters
<<interface>> :filtro
+ Filtro(List<InsertionBean>) : List<InsertionBean>
<<Boundary>>ModifyInsertionController
- vc : ViewController
- iv : insertionBean
- lg : languageFactory
+ setInsertion(InsertionBean) : void
+ modify() : void
+ delete() : void
FiltroAnd
- filtro Filtro
- altroFiltro : Filtro
+ Filtro(List<InsertionBean>) : List<InsertionBean>
FiltroBook
- book Boolean
+ Filtro(List<InsertionBean>) : List<InsertionBean>
FiltroNotes
- notes Boolean
+ Filtro(List<InsertionBean>) : List<InsertionBean>
<<Boundary>>ModifyInserMenuController
- vc : ViewController
- research : String
- lg : languageFactory
- filters : Filters
+ goBack() : void
+ setUser(Integer) : void
FiltroCity
- city String
+ Filtro(List<InsertionBean>) : List<InsertionBean>
FiltroUni
- uni String
+ Filtro(List<InsertionBean>) : List<InsertionBean>
FiltroSubj
- subj String
+ Filtro(List<InsertionBean>) : List<InsertionBean>


<<Entity>>Message
- sender : Integer[1]
- to : Integer[1]
- desc : String [50]
- date : Date[1]
+ Message(Integer,Integer,String,Date) : Message
<<Controller>>MessageController
- instance : MessageController
+ getMessageList(UserBean) : Message[]
+ getConversation(Integer) : Message[]
+ newMessage(Integer,Integer,String) : voic
+ getInstance() : MessageController
<<Boundary>>MessageMenuController
- vc : ViewController
- lg : languageFactory
+ update(UserBean) : void
+ goBack() : void
<<Boundary>>MessageDetailController
- vc : ViewController
- lb : UserBean
- lg : languageFactory
+ update(UserBean,Integer) : void
+ submit() : void
<<Boundary>>OrderDetailController
- vc : ViewController
- lg : languageFactory
+ update() :void
+ setOrder(Order) : void
+ pay() : void
<<Controller>>OrderController
- instance : OrderController
+ getList() : order[]
+ payOrder(Integer) : void
+ getOrdersInfo(String) : order[]
+ getOrderDetail(Integer id) : order
+ getInstance() : OrderController
<<Entity>> Order
- item : Item[1]
- id : Integer[1]
- seller : String[10]
- sold : Boolean[1]
- paid : Boolean[10]
+ Order() : Order
<<Entity>>Item
- nome : String [20]
- prezzo : Integer[1]
+ Item(String,Integer) : Item
<<Boundary>>OrderMenuViewController
- vc : ViewController
- lg : languageFactory
+ updateOrders(String) : void
+ back() : void


<<Boundary>>AdminMenuController
- vc : ViewController
+ update(UserBean) : void
+ logout() : void
<<Controller>>ReportController
+ newReport(Integer,String,Integer) : void
+ getReport() : List<Report>
+ ban(Integer,Integer) : void
+ removeReport(Integer) : void
<<Entity>>Report
- ID : integer[1]
- insId : Integer[1]
- Desc : String[300]
- reporter : integer[1]
+ getDescription(Integer) : String
+ newReport() : void
+ Report(Integer,Integer,String,Integer) : Report
<<Boundary>>ReportDetailController
- vc : ViewController
- lg : languageFactory
- lv : userBean
- rep : Report
+ noAction() : void
+ ban() : void
+ setReport(Report) : void
+ setUser(UserBean) : void
<<Entity>>User
- ID : integer[1]
- Name : String [10]
- Surname : String[10]
- isCompany : Boolean
- username : String[20]
- password : String[20]
- logo : Image[1]
+ Utente(String,String,String,String,Boolean,Integer,Image) : Utente
+ update() : boolean
+ tobean() : UserBean
<<Boundary>>LoginViewController
- vc : ViewController
- dispatch : String
- lg : LanguageFactory
+ setDispatch(String) : void
+ login() : void
<<Boundary>>ModifyProfileViewController
- vc : ViewController
- lb : userBean
- lg : LanguageFactory
+ submit() : void
+ goBack() : void
+ setLb(UserBean) : void
<<Controller>>LoginController
- instance : LoginController
+ validate(String,UserBean) : boolean
+ getUserFromId(Integer) : UserBean
+ getIdFromUsername(String) : Integer
+ updateInfo(UserBean) : boolean
+ login(String,String) : User
+ getInstance() : LoginController
<<Controller>>RegistrationController
- instance RegistrationController
+ getInstance() : RegistrationController
+ registraUtente(String,String,String,String,Boolean,String) : boolean
+ registraSocieta(String,String,String,Boolean,File) : boolean
<<Boundary>>RegisterMenuController
- vc : ViewController
- logo : File
- lg : LanguageFactory
+ submit() : void
+ upload() : void
+ studentSelected() : void
+ companySelected() : void



The ViewController organizes the creation of the GUI on the local JavaFX version of the application
every Boundary Controller sends a message to the controller to change interface, by setting a new scene or removing and setting a scene from the Deque (that is used as a stack) to go back.
Also the ViewController sends messages to the controllers if they need an attribute to be set.
I didn't use the association arrows because they would have made the document unreadable.

I used principally two design pattern : Singleton and Decorator.
Singleton was used on the controllers, because it reduces the occupation of the objects of type controller that would have been created, also for the local application we need the access the Deque in the viewController globally, so instead of usign a global variable we have just one access point, that is the ViewController instance, that contains the Deque we need to be able to go back between pages.

The second patter is decorator, used for filtering the research of an Insertion.
We got the Interface filtro that exposes the operation Filtro(), each Filtro implements the interface and ovverides the method, we then combine the Filters to get a filtered research.

We are also using DAOs to translate the entities into DB information and viceversa.

The document is for the local application in javaFX, but it also works if instead of a boundary controller we put a Servlet, since the Controller/Entity/Dao part of the application works independently from the kind of View we are using.