Universidade de Lisboa – Faculdade de Ciências

Construction of Software Systems

CSS Project
Part 2

May 8th 2022

—

Ana Luísa Patinho

Considerations

Business Layer

- All Bug fixes from the first part of the project, were made in the *git* repository of the first part of the project.
- The services for this goal were adapted to be session beans. They are all stateless session bean and have the @Stateless annotation.
- Each handler class has been assigned a session bean. The CreateActivityHandler handlers belong to the stateless session beans, however, the BuyMonthlyParticipationHandler, ScheduleOccasionalActivityHandler and SetNewScheduleHandler are session bean that store state, because to execute their operations it is necessary to perform several validations and user interactions.
- Communication between the database is no longer dependent on EntityManagers created in each handler class, these transactions have been replaced in the classes representing the catalogs (also stateless session beans) with a single instance of an EntityManager object due to the @PersistenceContext annotation. Each method that persists an object in the database is annotated with @Transactional(Transactional.TxType.REQUIRES_NEW) if the object was new to the database or just @Transactional.
- For transferring the information needed to show the user and protect the integrity of the remotely sent data, the Data Transfer Object standard was used. For each method of a handler with a more sophisticated return object type different serializable DTO objects were created with attributes with simpler data types.
- At the concurrency level, our group defined attributes with the @Version annotation in four classes, namely Activity, Reservation, Schedule and Session, because they are responsible for creating the different objects and relationships in the different use cases.

Considerations

WEB Client

In developing the web client we used the interfaces annotated with @Remote to perform the use cases Buy Monthly Participation in Regulae Activity and Schedule Occasional Activity. In order to make these interfaces useful the Front Controller pattern was used in which different Action classes are used. The correspondence between the web addresses and the Action classes is maintained in the app.properties file which contains the following URL's:

```
appRoot/action/participation/newParticipation=java:module/NewBuyAction
appRoot/action/participation/chooseActivity=java:module/ChooseActivityAction
appRoot/action/participation/chooseSchedule=java\:module/ChooseScheduleAction
appRoot/action/participation/finalizeBuying=java\:module/CompleteBuyingAction
appRoot/action/schedule/occasionalSchedule=java:module/NewScheduleOccasionalAction
appRoot/action/schedule/chooseSpecialty=java\:module/ChooseSpecialtyAction
appRoot/action/schedule/chooseOccasional=java\:module/ChooseOccasionalAction
appRoot/action/schedule/chooseDatesTimes=java\:module/ChooseDatesTimesAction
appRoot/action/schedule/chooseInstructor=java\:module/ChooseInstructor
```

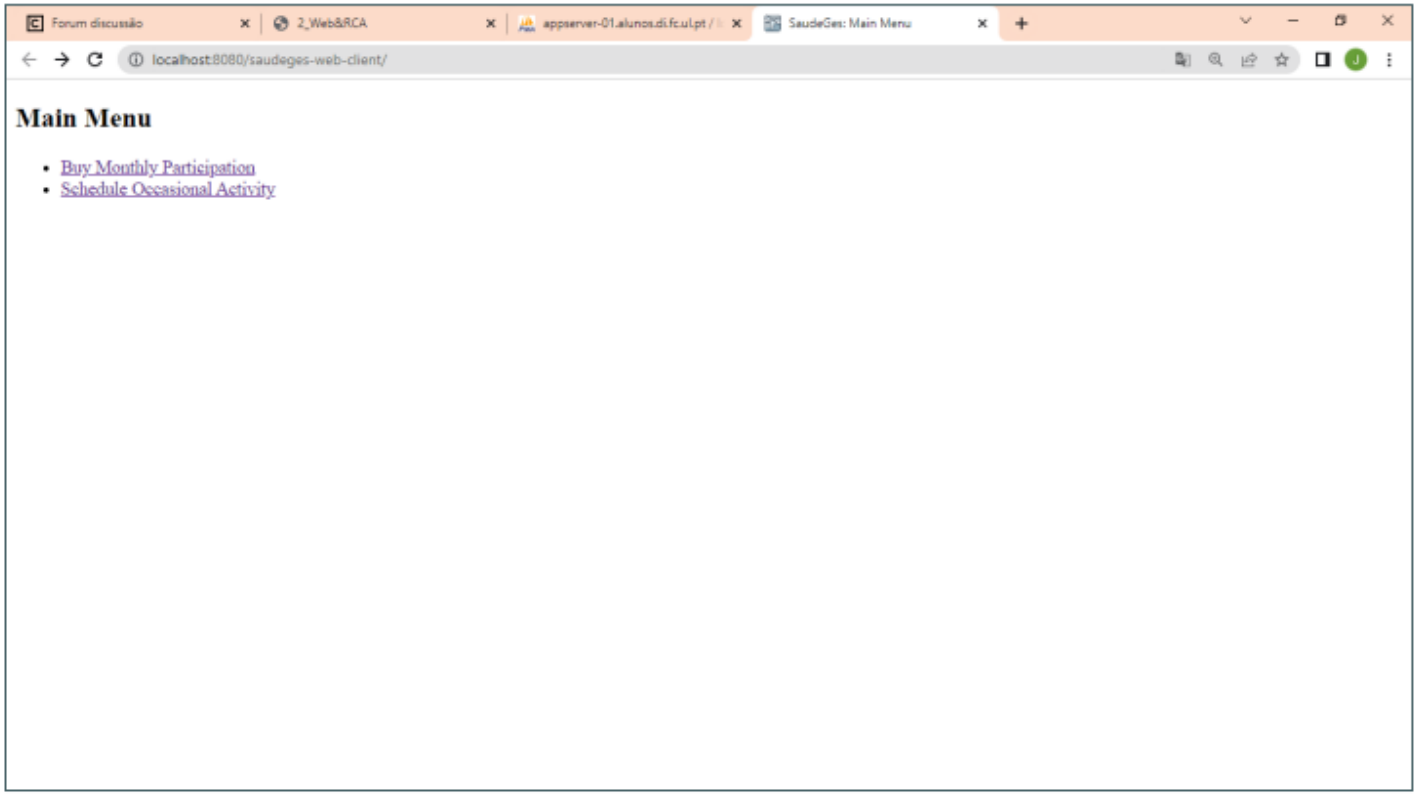
The view is maintained with JSP files that use the different Actions to execute the various functionalities of the application.

GUI Client

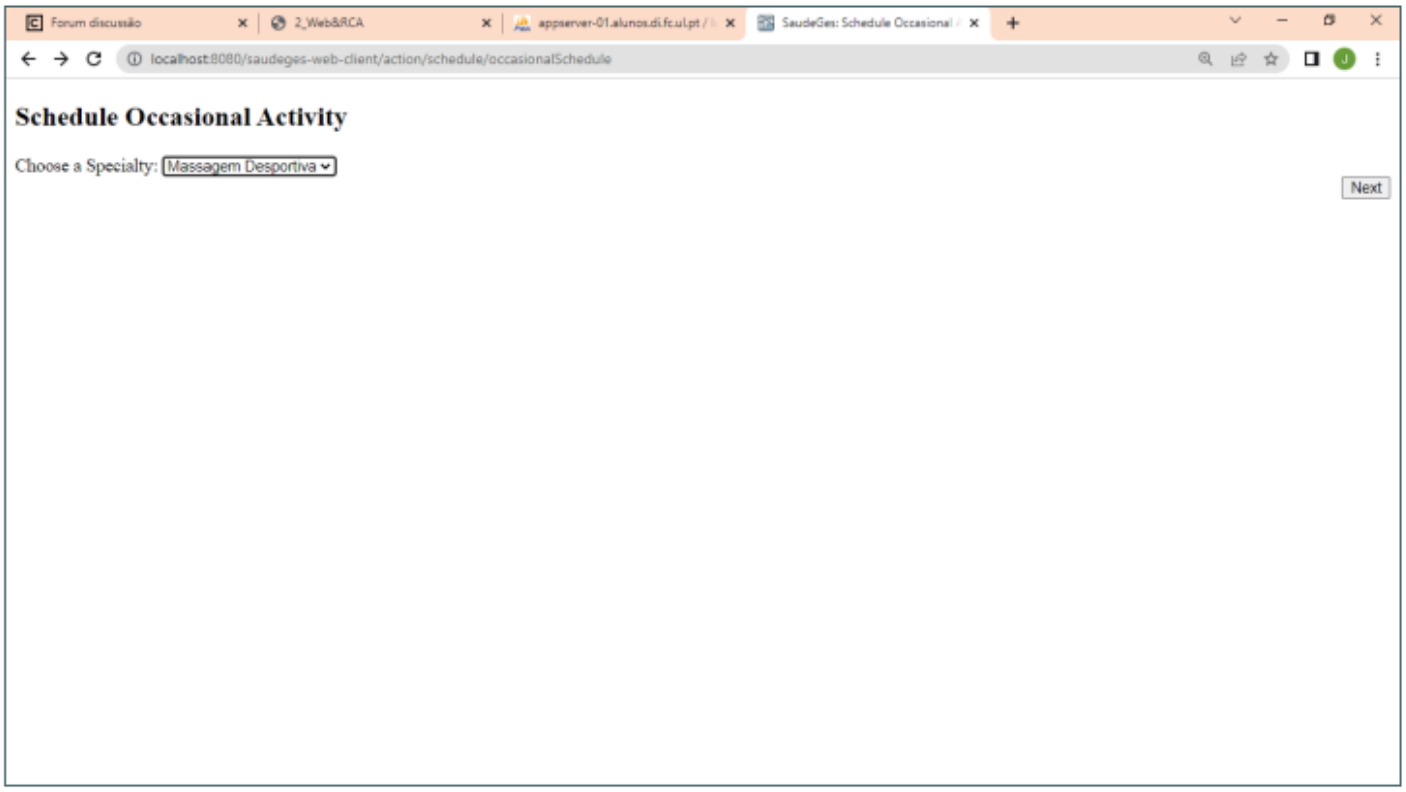
In the GUI client project five screens were created: main menu (main.fxml), create activity (createActivity.fxml), select regular activity (selectRegularActivity.fxml), set schedule (setSessions.fxml) and choose instructor (setInstructor.fxml). The main menu is just for the user to choose which use case they want to run. Each screen responsible for one use case and uses the remote session beans to execute its functionality through the controller classes. To establish the connection between the controller and the view (.fxml) the Model classes were implemented for each object.

Some Ilustrations

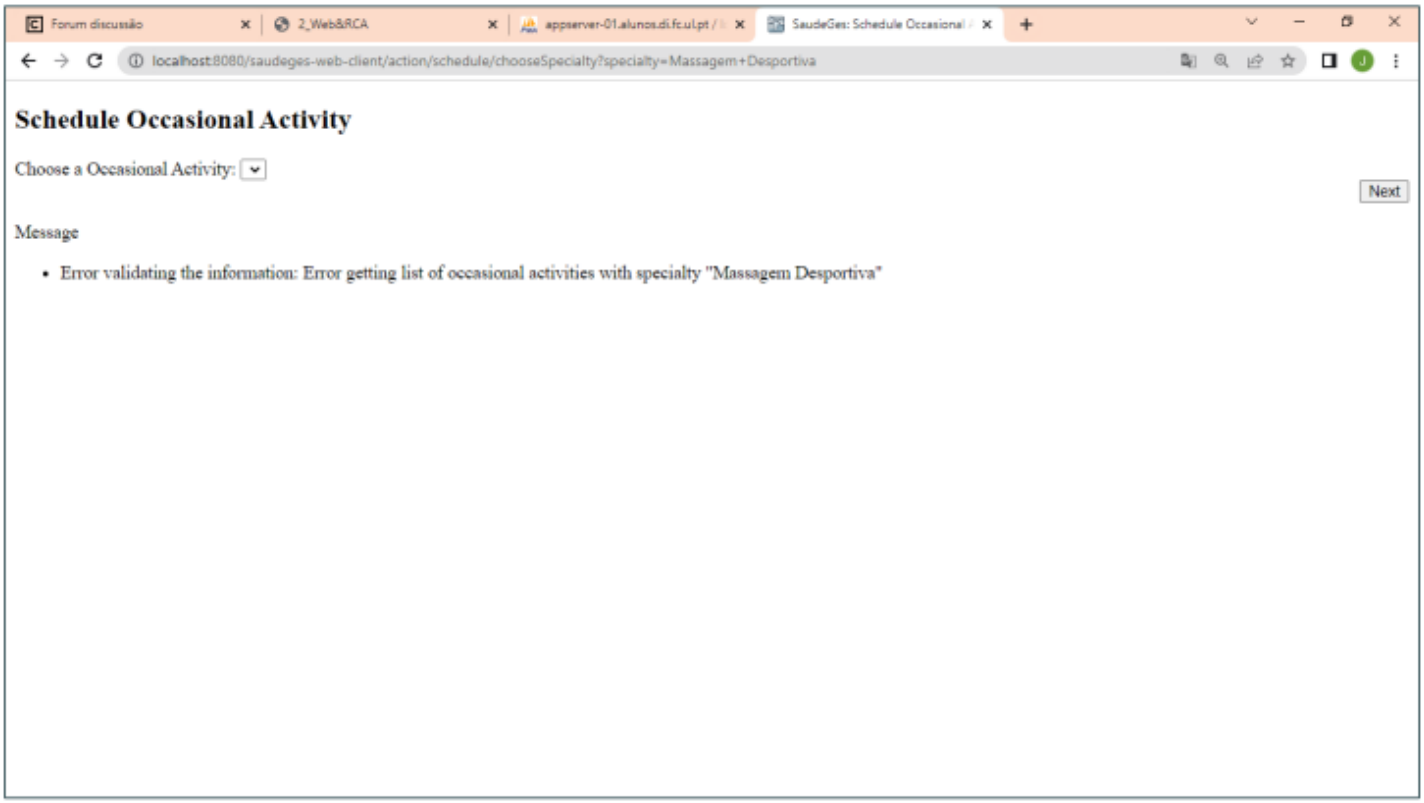
WEBClient Menu



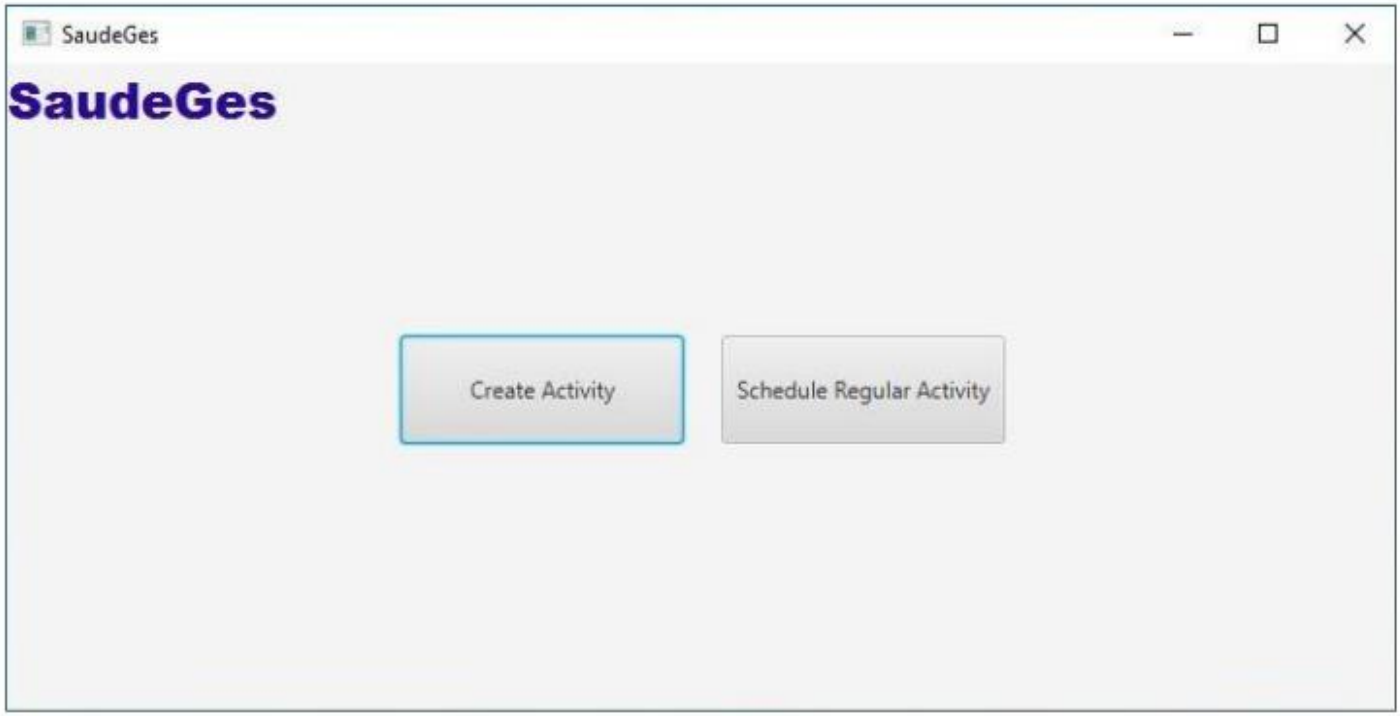
WEBClient ScheduleOccasionalActivity



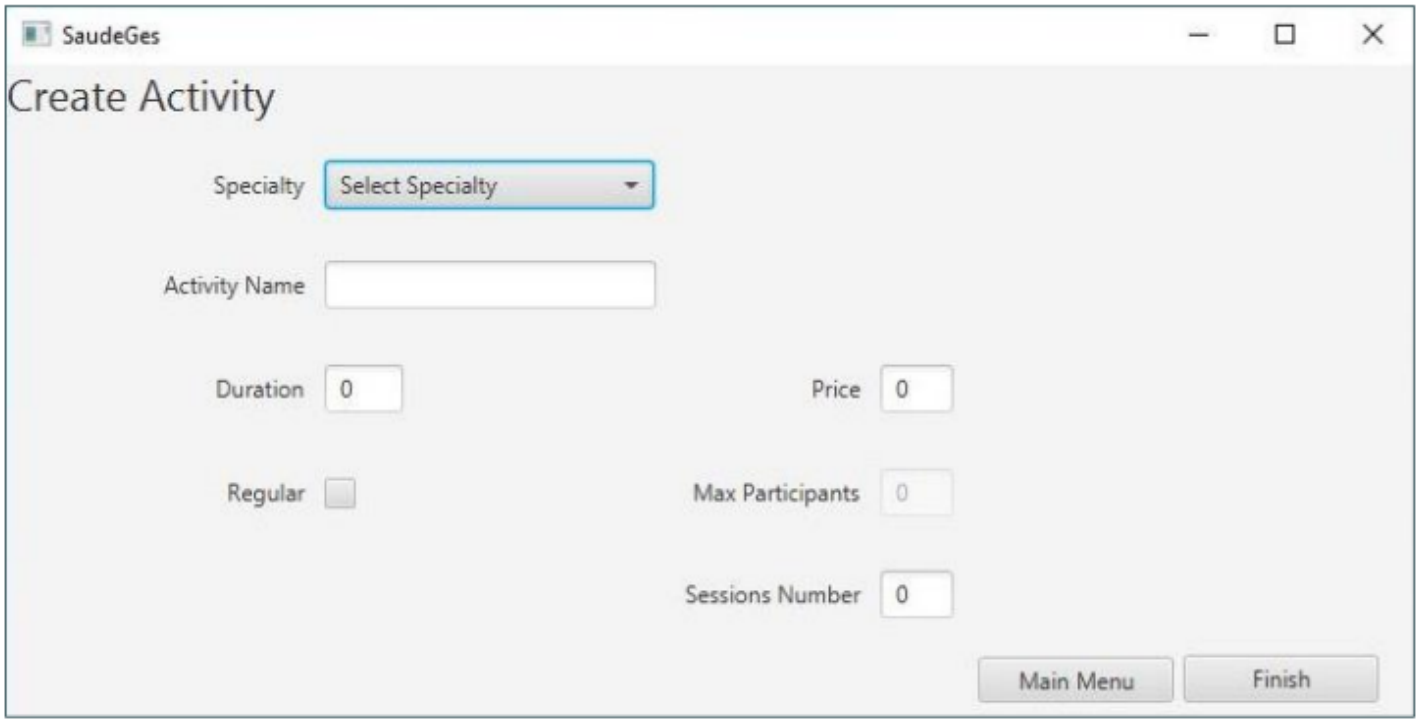
WEBClient ScheduleOcasionalActivity Error



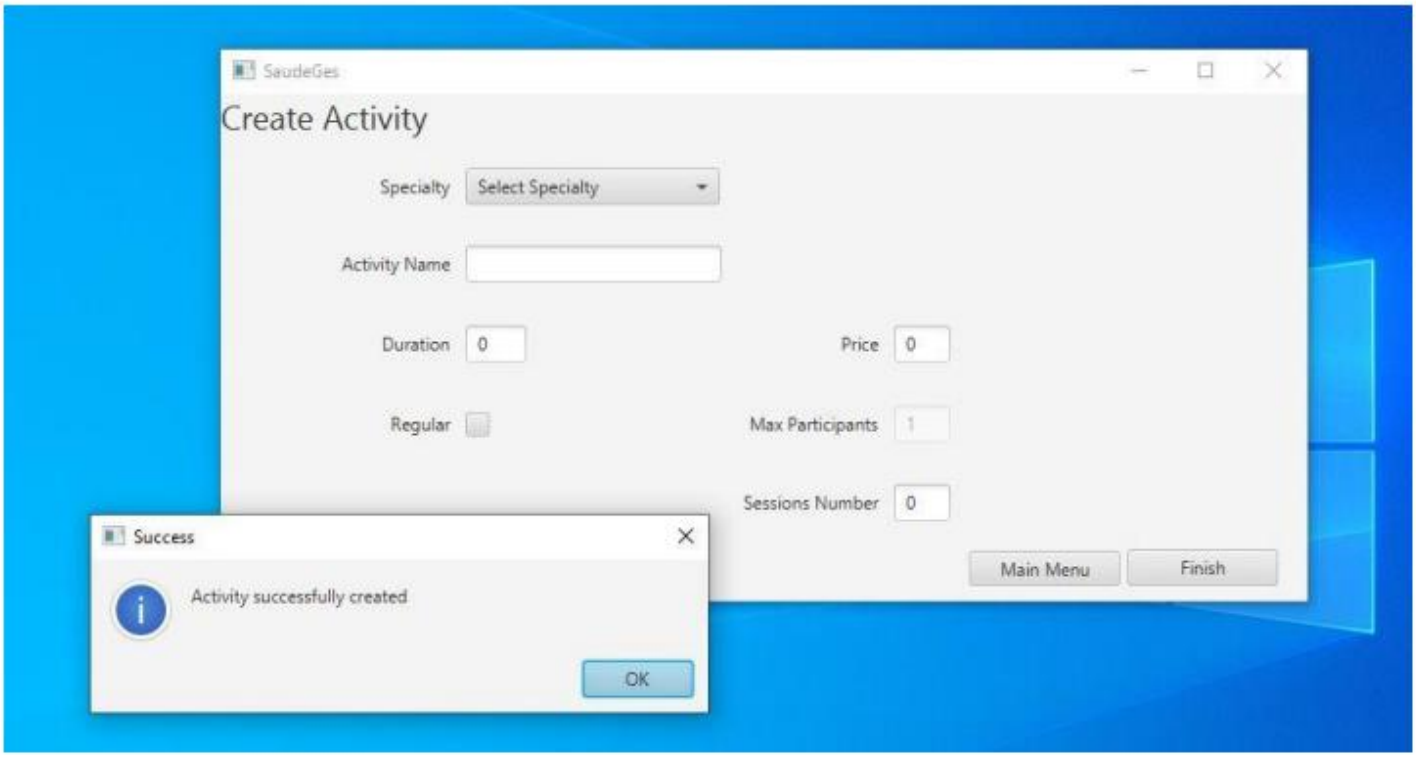
GUIClient Menu



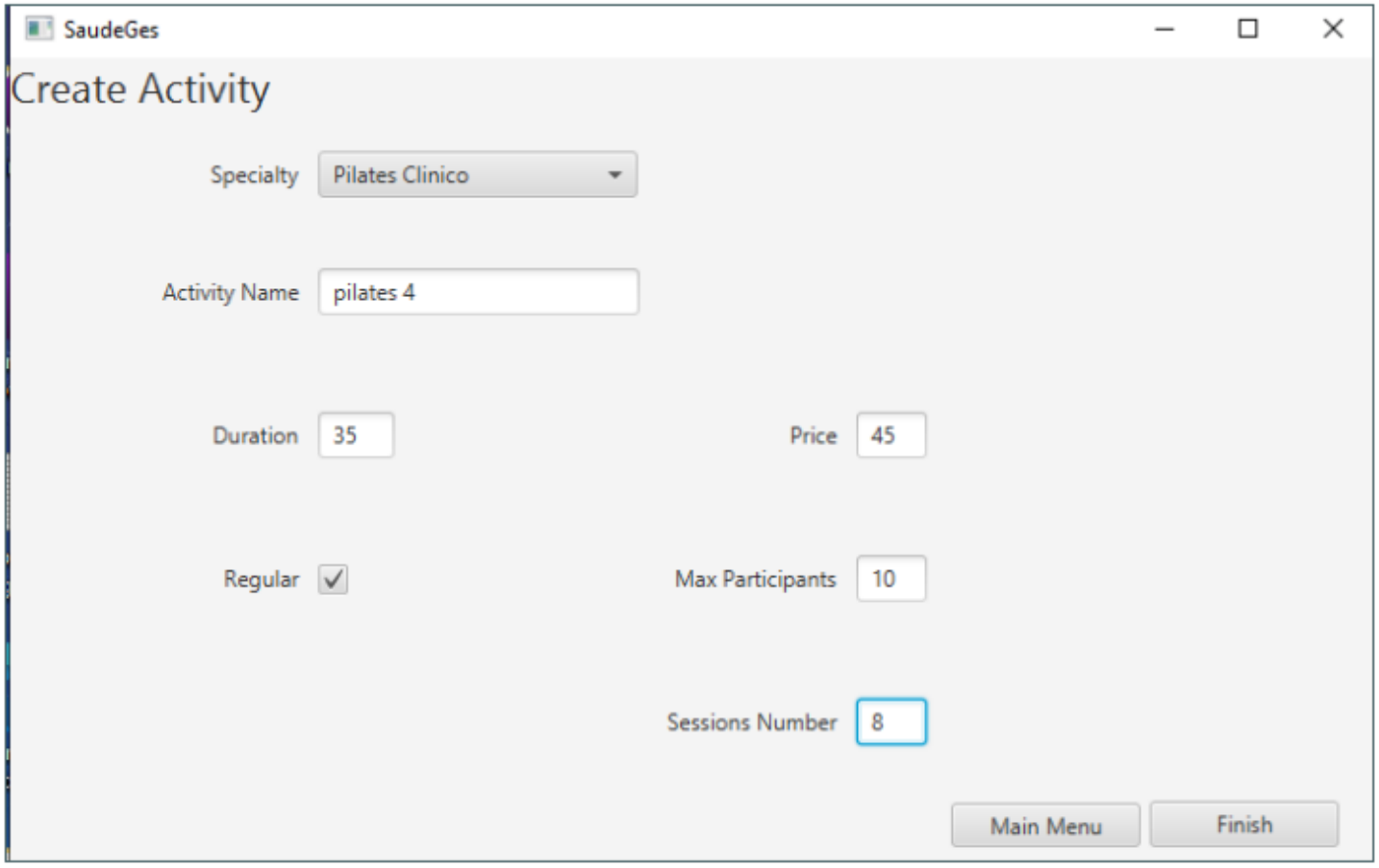
GUIClient CreateActivity



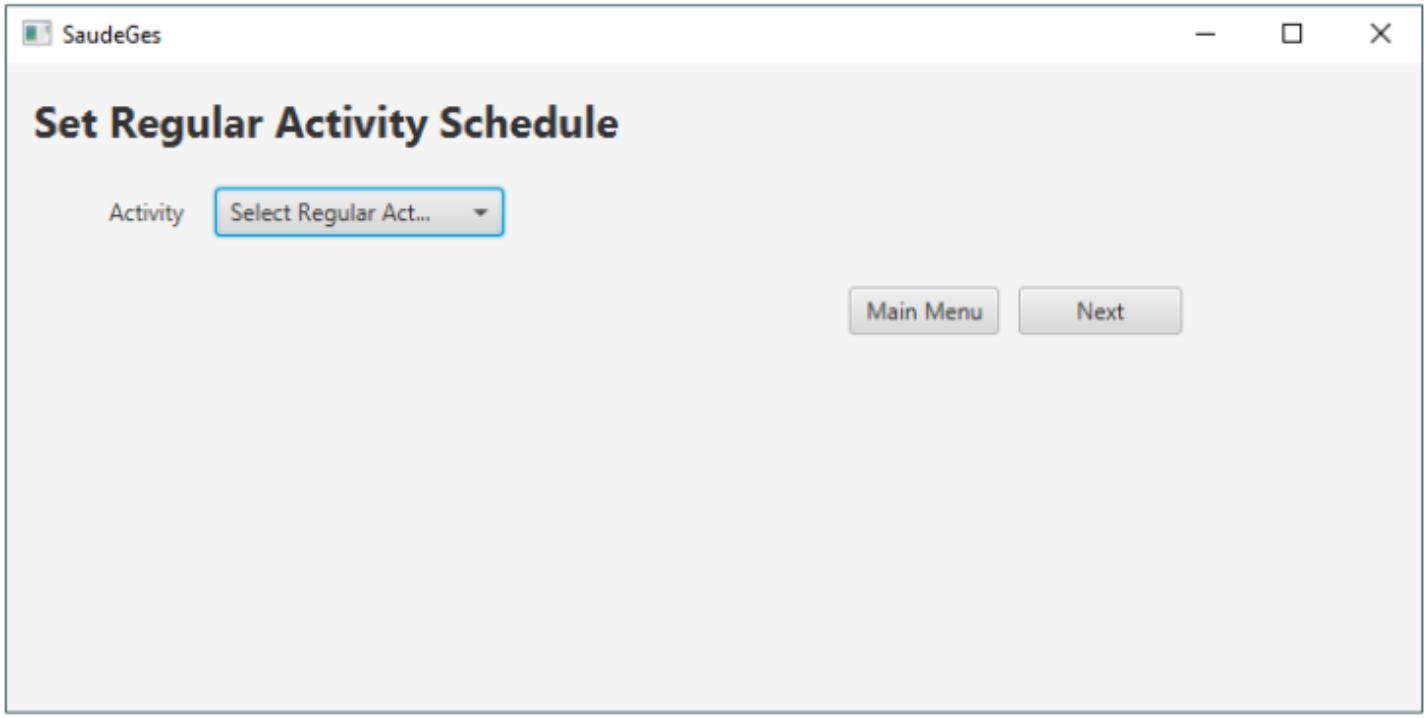
GUIClient CreateActivity Successful



GUIClient CreateRegularActivity



GUIClient SetRegularActivitySchedule



GUIClient CreateOccasionalActivity Successful