SOFTWARE REQUIREMENTS SPECIFICATION

1. INTRODUCTION

1.1. AIM OF THE DOCUMENT

This document describes the "*FitApp*" system, what functionality it provides to users and that they expect from it. It contains information that can help developers and testers during their job.

Functionalities that are written in gray are not implemented.

1.2. OVERVIEW OF THE DEFINED SYSTEM

The system is an application that allows customer to keep contact with their favourites gyms.

The system defines two type of users and one of them has two type of access to the system:

1. *Unregistered Users*: they cannot perform any operation until they complete the sign-up process.
2. *Registered Users:* there are three types of Registered Users.
   a. *Guest:* a guest user represents the users that are trying to sign-up into the system but does not filled all the required information about them.
   b. *User*: this represents a gym's customer in the real world. It can book for a training session specifying the timing of its training and the distance that it wants to reach. It also can write reviews on sessions that it booked to increase or decrease gym's or trainer's rank.
   c. *Gym Manager*: a gym manager is the person that administrate a gym and its trainers. It can add or remove a trainer and edit the list of courses that it can do. It can create training sessions specifying the timing, if a session is individual, which trainer takes the course and can add a short description to it.

1.3. OPERATIONAL SETTINGS

The system is built within the Java ecosystem (Java8), using JavaFX as graphic frameworks.

Users can interact with the system using the standalone version or its related web-app, built with JSP and Servlet technologies and Tomcat as server.

1.4. RELATED SYSTEMS, PROS AND CONS

Possible related systems are gym's personal apps, built for specific gym or gym franchising. Most of them allow customer to book for training session or manage their training sheet. And, generally, they can manage users' data better because they collect more information about them. This system instead, allows users to search for registered gyms and their courses within the selected radius, reducing the distance between gyms and possible new customers.

Another related system is Google Maps. This Google's service is used by our system to make research on web and show the result with markers on the map. That kind of research is replicable with Maps, but our system allows users to book a training session with few clicks, without directly contact the interested gym.

2. USER STORIES

ANDREA EFFICACE:

- As a User, I want to manage my booked session, so that I can decide to delete or modify a training session.
- As a Gym Manager, I want to manage my personal trainers list, so that I can edit their course list, add new trainers, or delete them from the list.
- As a User, I want to write reviews for session I have booked, so that I can provide feedback for other users.

LORENZO ROSSI:

- As a User, I want to find training sessions inside specified distance radius (in Km) in Google Map, so that I can train at gyms around me.
- As a Gym manager, I want to add new gym training sessions on monthly calendar setting: date, time, course, and trainer, so that I can schedule gym's monthly activities.
- As a Gym's manager user, I want to view gym session review, so that I can receive feedback from other users.

3. FUNCTIONAL REQUIREMENTS

ANDREA EFFICACE:

- The system shall provide a booking form that requires:
  - Time Start and Time End of a training session.
  - The day of the training session.
  - The maximum radius of distance, between user position and gyms.
- The system shall provide a User personal calendar that allow booked session management.
- The system shall provide a registration form for the Unregistered Users that asks for:
  - User's name in the system.
  - User's email.
  - User's password. This field must be confirmed with double insert of the password.
  - User's street.
  - If It wants to register into the system as Manager. *

**if the User is a manager, the form must include fields for the Gym's Name and the Gym's Street.*

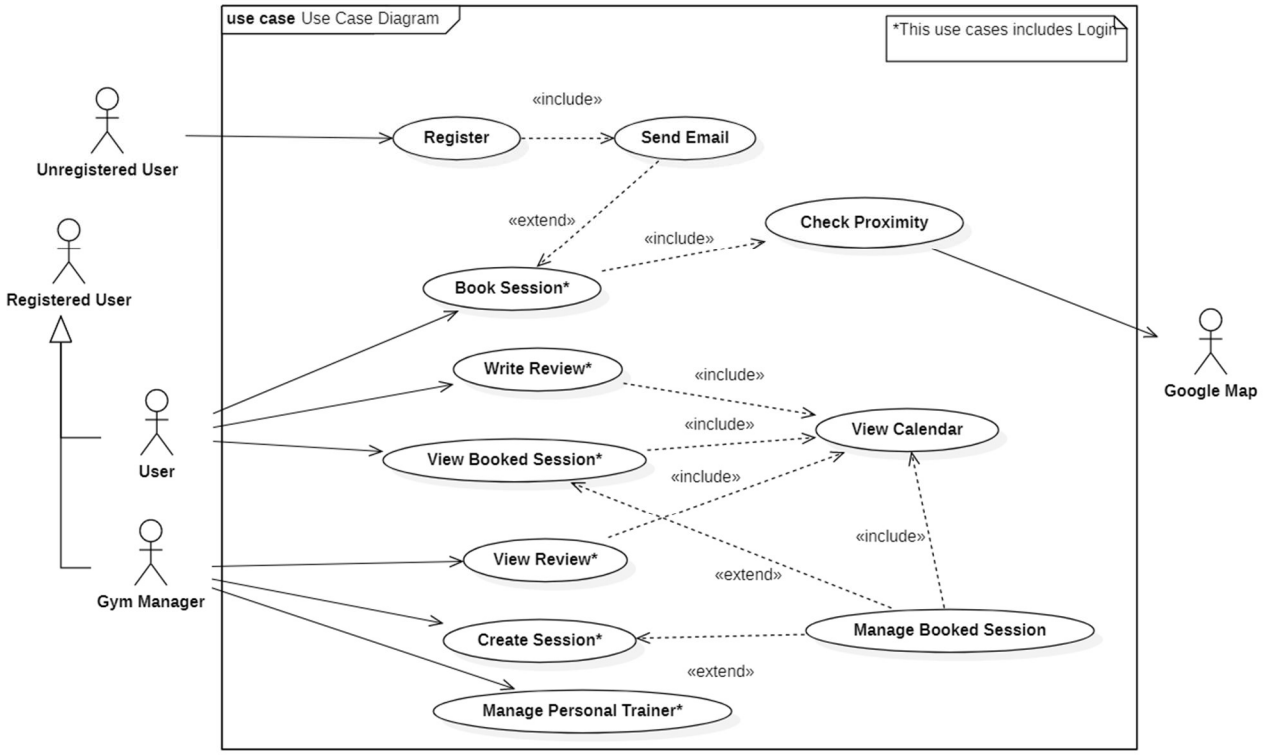
(not to be examined):

- The system shall provide a unique point of access for the three types of Registered Users
- The system shall provide a ranking system based on users' reviews.

LORENZO ROSSI:

- The system shall allow Users to search gym training sessions specifying the maximum distance radius (in Km) from their position.
- The system shall allow Users to delete booked sessions.
- The system shall provide an email platform between registered user and training session gym manager.

4. USE CASES OVERVIEW DIAGRAM

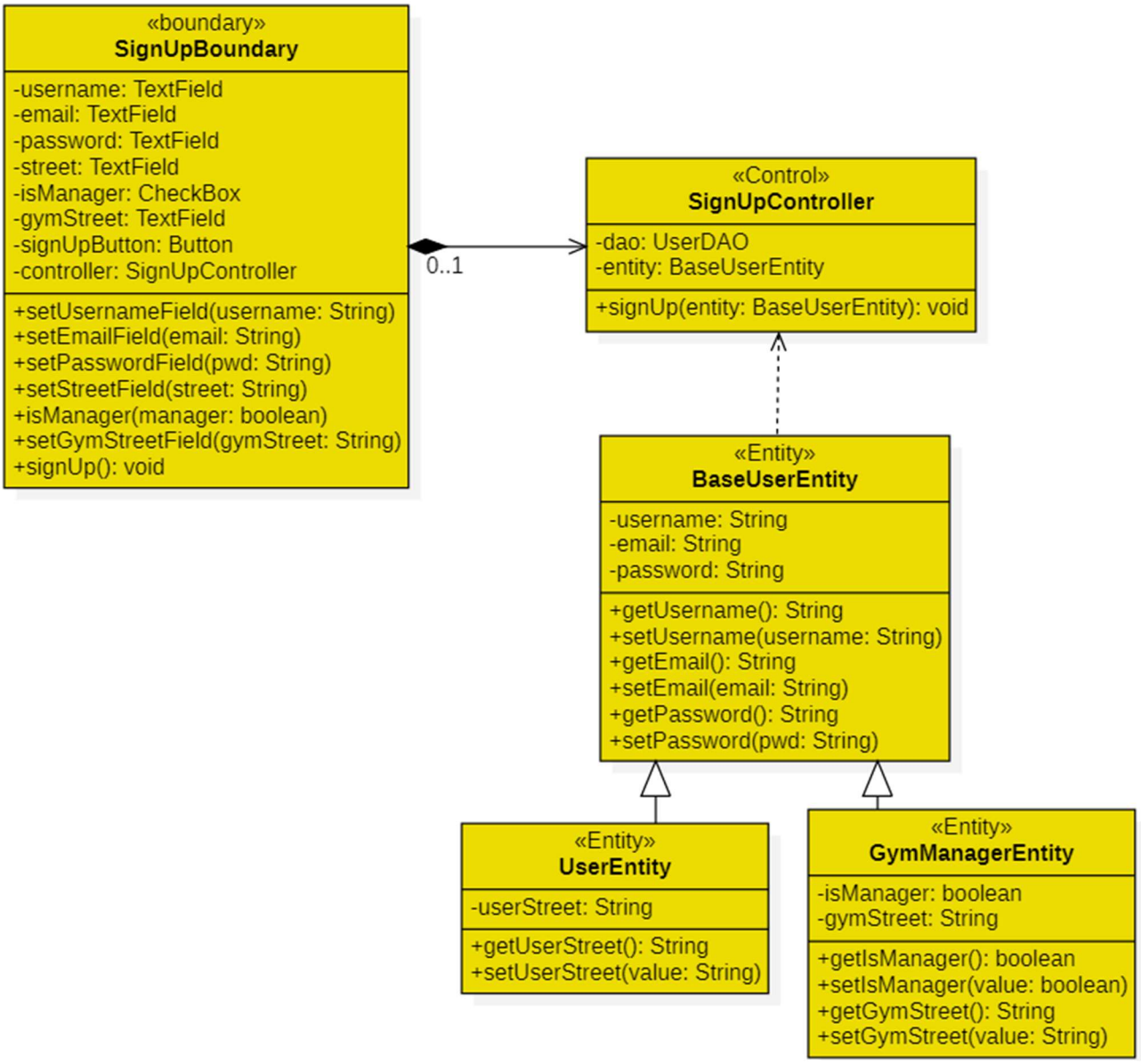


DESIGN

1. CLASS DIAGRAMS

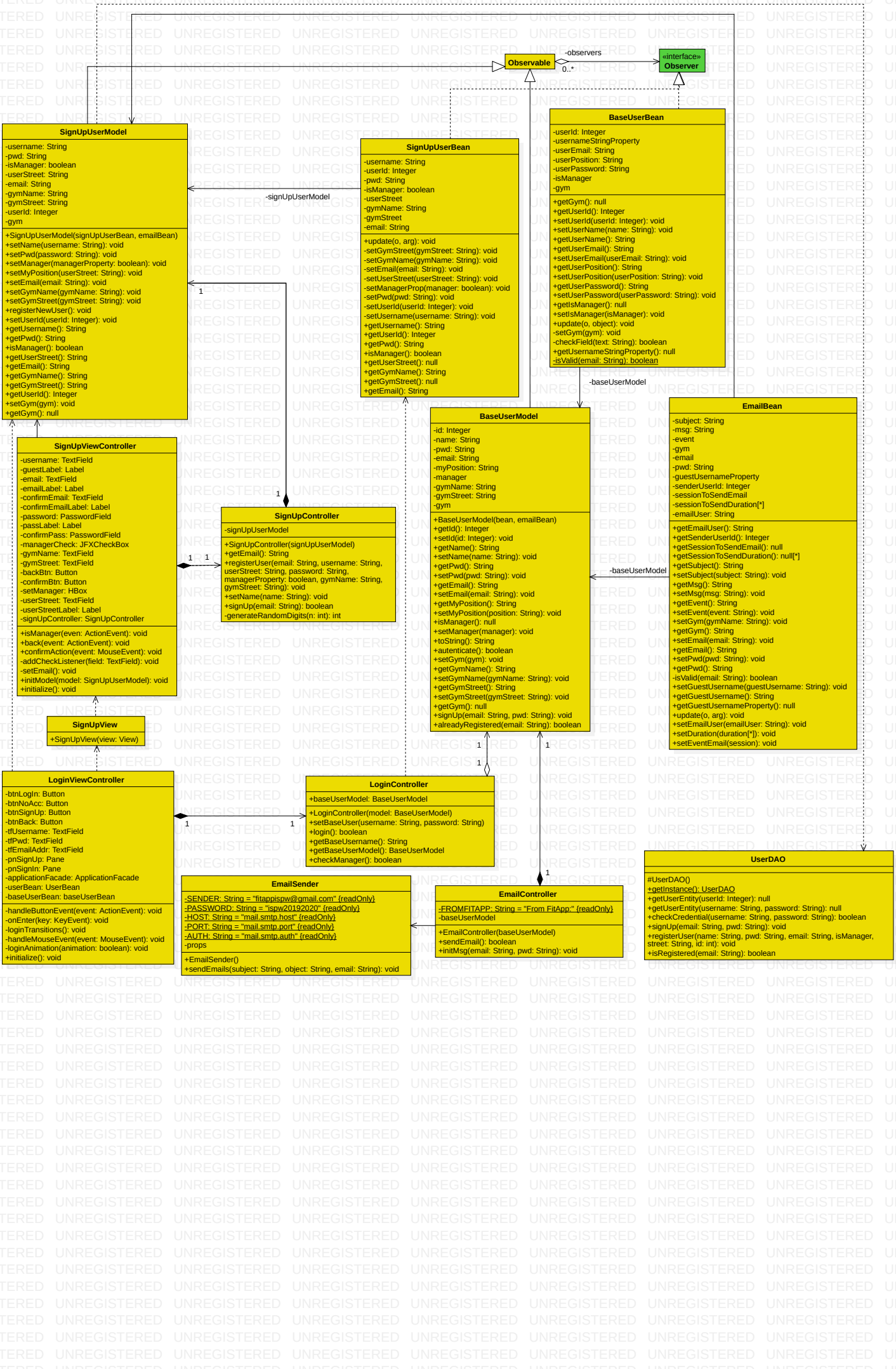
ANDREA EFFICACE

OLD



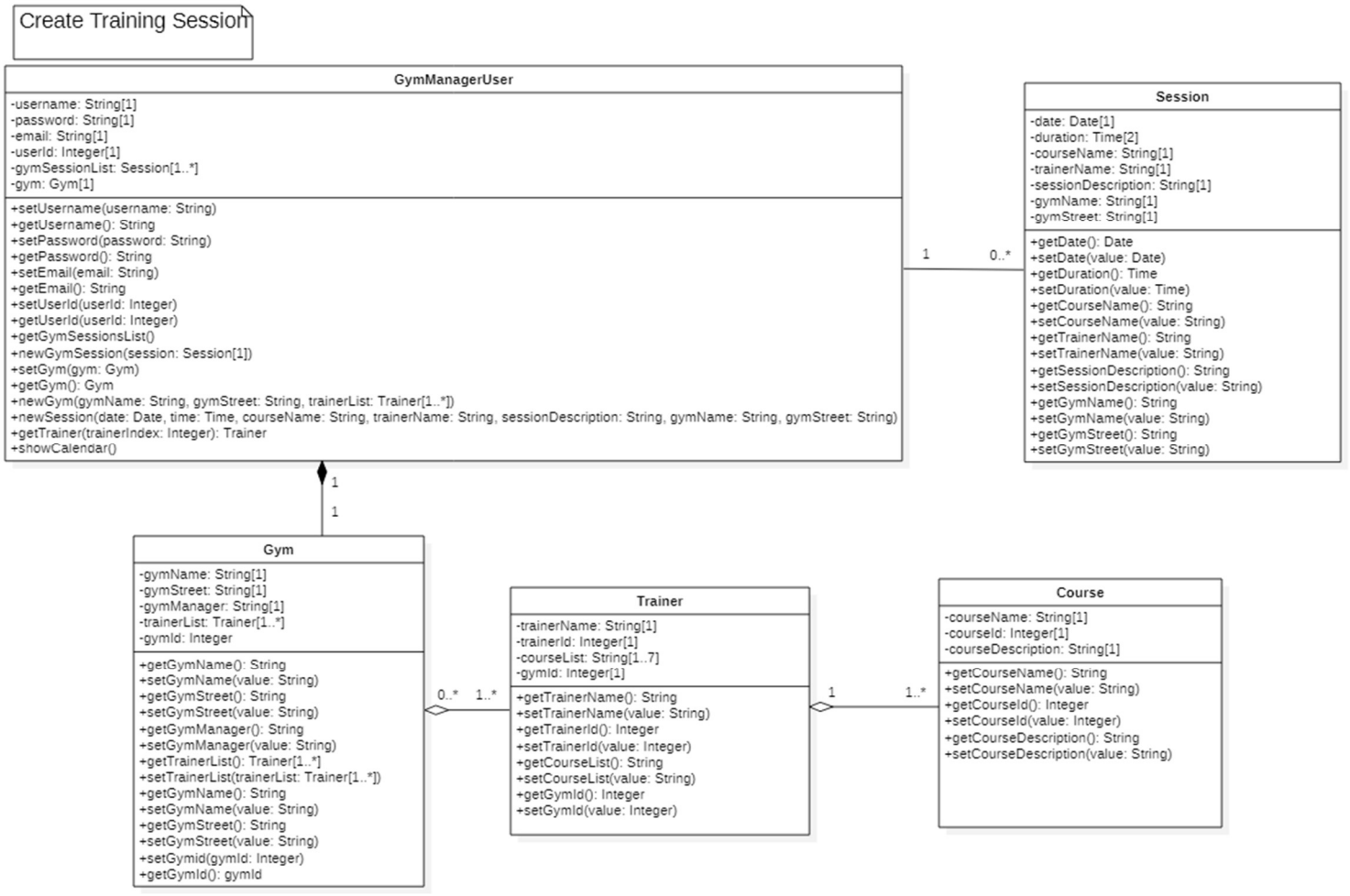
NEW

(on next page)


Observable
-observers
0..*
«interface»
Observer
SignUpUserModel
-username: String
-pwd: String
-isManager: boolean
-userStreet: String
-email: String
-gymName: String
-gymStreet: String
-userId: Integer
-gym
+SignUpUserModel(signUpUserBean, emailBean)
+setName(username: String): void
+setPwd(password: String): void
+setManager(managerProperty: boolean): void
+setMyPosition(userStreet: String): void
+setEmail(email: String): void
+setGymName(gymName: String): void
+setGymStreet(gymStreet: String): void
+registerNewUser(): void
+setUserId(userId: Integer): void
+getUsername(): String
+getPwd(): String
+isManager(): boolean
+getUserStreet(): String
+getEmail(): String
+getGymName(): String
+getGymStreet(): String
+getUserId(): Integer
+setGym(gym): void
+getGym(): null
SignUpUserBean
-username: String
-userId: Integer
-pwd: String
-isManager: boolean
-userStreet
-gymName: String
-gymStreet
-email: String
+update(o, arg): void
-setGymStreet(gymStreet: String): void
-setGymName(gymName: String): void
-setEmail(email: String): void
-setUserStreet(userStreet: String): void
-setManagerProp(manager: boolean): void
-setPwd(pwd: String): void
-setUserId(userId: Integer): void
-setUsername(username: String): void
+getUsername(): String
+getUserId(): Integer
+getPwd(): String
+isManager(): boolean
+getUserStreet(): null
+getGymName(): String
+getGymStreet(): null
+getEmail(): String
-signUpUserModel
1
BaseUserBean
-userId: Integer
-usernameStringProperty
-userEmail: String
-userPosition: String
-userPassword: String
-isManager
-gym
+getGym(): null
+getUserId(): Integer
+setUserId(userId: Integer): void
+setUserName(name: String): void
+getUserName(): String
+getUserEmail(): String
+setUserEmail(userEmail: String): void
+getUserPosition(): String
+setUserPosition(userPosition: String): void
+getUserPassword(): String
+setUserPassword(userPassword: String): void
+getIsManager(): null
+setIsManager(isManager): void
+update(o, object): void
-setGym(gym): void
-checkField(text: String): boolean
+getUsernameStringProperty(): null
-isValid(email: String): boolean
-baseUserModel
SignUpViewController
-username: TextField
-guestLabel: Label
-email: TextField
-emailLabel: Label
-confirmEmail: TextField
-confirmEmailLabel: Label
-password: PasswordField
-passLabel: Label
-confirmPass: PasswordField
-managerCheck: JFXCheckBox
-gymName: TextField
-gymStreet: TextField
-backBtn: Button
-confirmBtn: Button
-setManager: HBox
-userStreet: TextField
-userStreetLabel: Label
-signUpController: SignUpController
+isManager(even: ActionEvent): void
+back(event: ActionEvent): void
+confirmAction(event: MouseEvent): void
-addCheckListener(field: TextField): void
-setEmail(): void
+initModel(model: SignUpUserModel): void
+initialize(): void
SignUpController
-signUpUserModel
+SignUpController(signUpUserModel)
+getEmail(): String
+registerUser(email: String, username: String, userStreet: String, password: String, managerProperty: boolean, gymName: String, gymStreet: String): void
+setName(name: String): void
+signUp(email: String): boolean
-generateRandomDigits(n: int): int
1
1
1
BaseUserModel
-id: Integer
-name: String
-pwd: String
-email: String
-myPosition: String
-manager
-gymName: String
-gymStreet: String
-gym
+BaseUserModel(bean, emailBean)
+getId(): Integer
+setId(id: Integer): void
+getName(): String
+setName(name: String): void
+getPwd(): String
+setPwd(pwd: String): void
+getEmail(): String
+setEmail(email: String): void
+getMyPosition(): String
+setMyPosition(position: String): void
+isManager(): null
+setManager(manager): void
+toString(): String
+autenticate(): boolean
+setGym(gym): void
+getGymName(): String
+setGymName(gymName: String): void
+getGymStreet(): String
+setGymStreet(gymStreet: String): void
+getGym(): null
+signUp(email: String, pwd: String): void
+alreadyRegistered(email: String): boolean
-baseUserModel
EmailBean
-subject: String
-msg: String
-event
-gym
-email
-pwd: String
-guestUsernameProperty
-senderUserId: Integer
-sessionToSendEmail
-sessionToSendDuration[*]
-emailUser: String
+getEmailUser(): String
+getSenderUserId(): Integer
+getSessionToSendEmail(): null
+getSessionToSendDuration(): null[*]
+getSubject(): String
+setSubject(subject: String): void
+getMsg(): String
+setMsg(msg: String): void
+getEvent(): String
+setEvent(event: String): void
+setGym(gymName: String): void
+getGym(): String
+setEmail(email: String): void
+getEmail(): String
+setPwd(pwd: String): void
+getPwd(): String
-isValid(email: String): boolean
+setGuestUsername(guestUsername: String): void
+getGuestUsername(): String
+getGuestUsernameProperty(): null
+update(o, arg): void
+setEmailUser(emailUser: String): void
+setDuration(duration[*]): void
+setEventEmail(session): void
SignUpView
+SignUpView(view: View)
LoginViewController
-btnLogIn: Button
-btnNoAcc: Button
-btnSignUp: Button
-btnBack: Button
-tfUsername: TextField
-tfPwd: TextField
-tfEmailAddr: TextField
-pnSignUp: Pane
-pnSignIn: Pane
-applicationFacade: ApplicationFacade
-userBean: UserBean
-baseUserBean: baseUserBean
-handleButtonEvent(event: ActionEvent): void
-onEnter(key: KeyEvent): void
-loginTransitions(): void
-handleMouseEvent(event: MouseEvent): void
-loginAnimation(animation: boolean): void
+initialize(): void
LoginController
+baseUserModel: BaseUserModel
+LoginController(model: BaseUserModel)
+setBaseUser(username: String, password: String)
+login(): boolean
+getBaseUsername(): String
+getBaseUserModel(): BaseUserModel
+checkManager(): boolean
1
1
1
1
1
1
EmailSender
-SENDER: String = "fitappispw@gmail.com" {readOnly}
-PASSWORD: String = "ispw20192020" {readOnly}
-HOST: String = "mail.smtp.host" {readOnly}
-PORT: String = "mail.smtp.port" {readOnly}
-AUTH: String = "mail.smtp.auth" {readOnly}
-props
+EmailSender()
+sendEmails(subject: String, object: String, email: String): void
EmailController
-FROMFITAPP: String = "From FitApp:" {readOnly}
-baseUserModel
+EmailController(baseUserModel)
+sendEmail(): boolean
+initMsg(email: String, pwd: String): void
UserDAO
#UserDAO()
+getInstance(): UserDAO
+getUserEntity(userId: Integer): null
+getUserEntity(username: String, password: String): null
+checkCredential(username: String, password: String): boolean
+signUp(email: String, pwd: String): void
+registerUser(name: String, pwd: String, email: String, isManager, street: String, id: int): void
+isRegistered(email: String): boolean

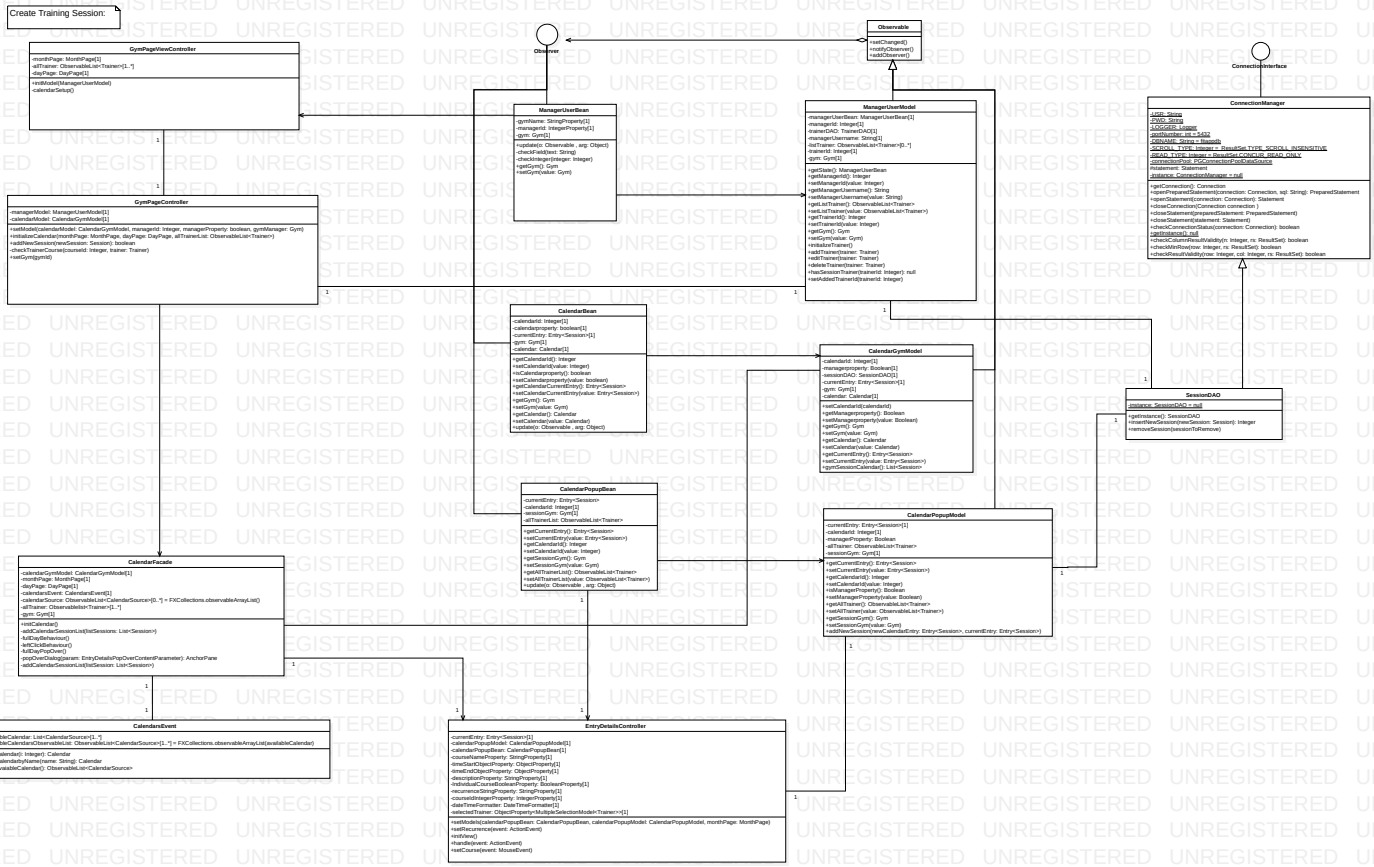
LORENZO ROSSI

OLD



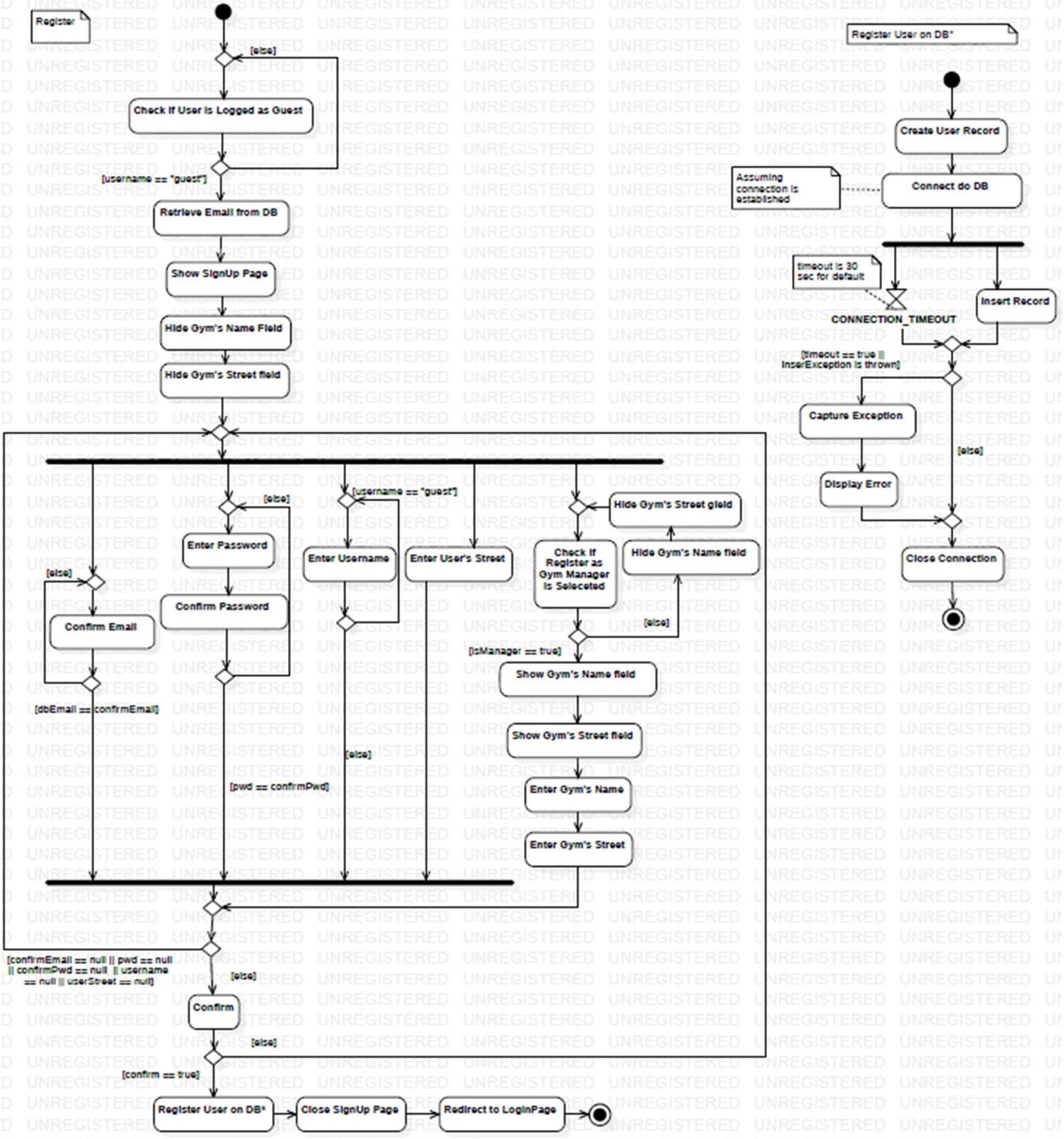
NEW

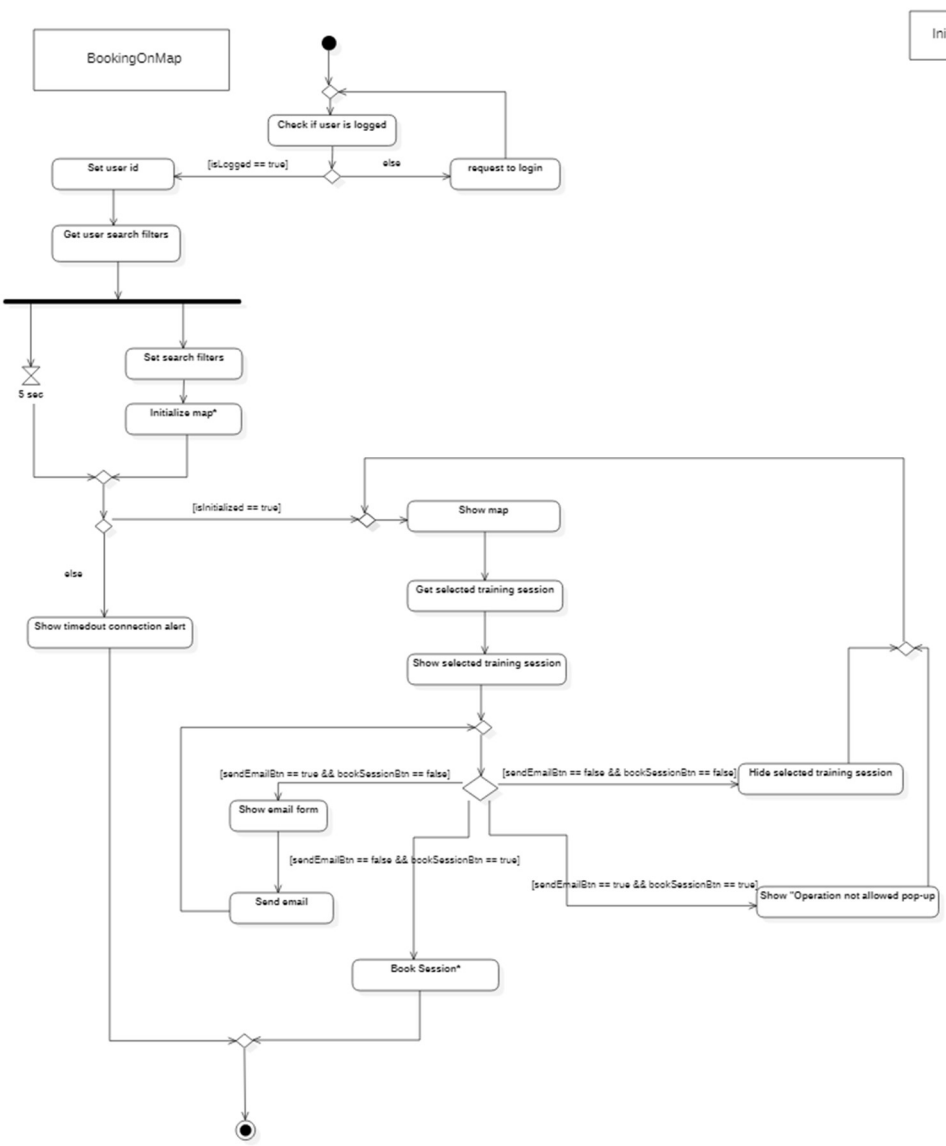
(on next page)

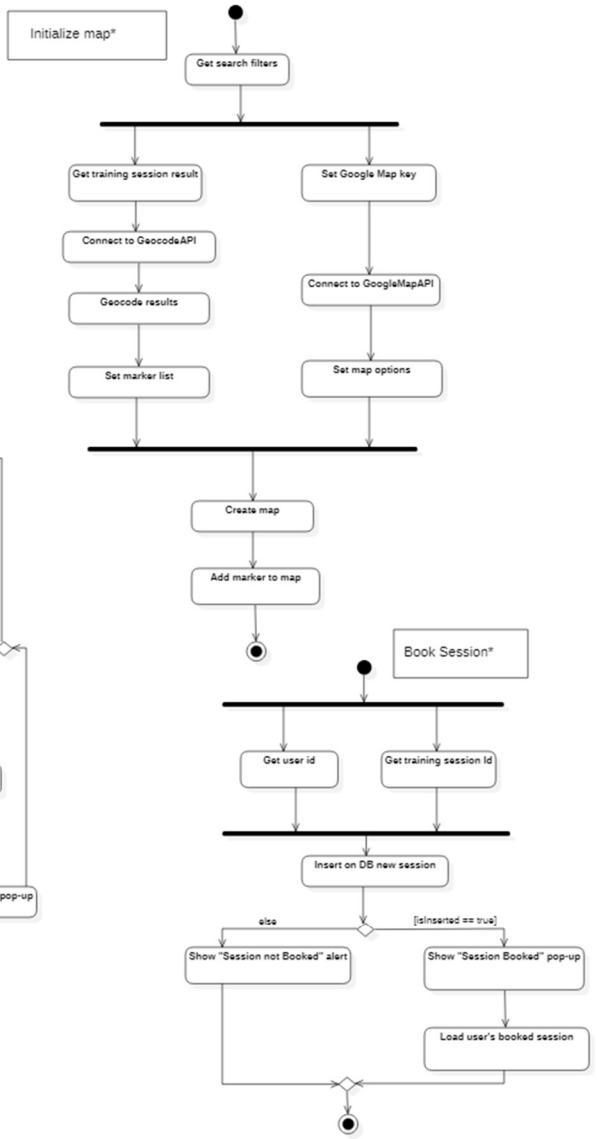

Create Training Session:
GymPageViewController
-monthPage: MonthPage[1]
-allTrainer: ObservableList<Trainer>[1..*]
-dayPage: DayPage[1]
+initModel(ManagerUserModel)
-calendarSetup()
Observer
Observable
+setChanges()
+notifyObservers()
+addObserver()
ConnectionInterface
ManagerUserBean
-gymName: StringProperty[1]
-managerId: IntegerProperty[1]
-gym: Gym[1]
+update(o: Observable , arg: Object)
-checkEmptyString()
-checkInteger(integer: Integer)
+getGym(): Gym
+setGym(value: Gym)
ManagerUserModel
+managerUserBean: ManagerUserBean[1]
+managerId: Integer[1]
+trainerDAO: TrainerDAO[1]
+managerUsername: String[1]
-listTrainer: ObservableList<Trainer>[0..*]
+monthPage[1]
-gym: Gym[1]
+getState(): ManagerUserBean
+getManagerId(): Integer
+setManagerId(value: Integer)
+getManagerUsername(): String
+setManagerUsername(value: String)
+getListTrainer(): ObservableList<Trainer>
+setListTrainer(value: ObservableList<Trainer>)
+getTrainerId(): Integer
+setTrainerId(value: Integer)
+getGym(): Gym
+setGym(value: Gym)
+initializeTrainer()
+addTrainer(trainer: Trainer)
-addTrainer(trainer: Trainer)
+deleteTrainer(trainer: Trainer)
+hasSessionTrainer(trainerId: Integer): null
+setAddedTrainerId(trainerId: Integer)
ConnectionManager
-LOGIN: String
-PWD: String
-LOGGER: Logger
-portNumber: int = 5432
-DBNAME: String = 'fitapp3b'
-SCROLL_TYPE: Integer = ResultSet.TYPE_SCROLL_INSENSITIVE
-READ_TYPE: Integer = ResultSet.CONCUR_READ_ONLY
-connectionPool: PGConnectionPoolDataSource
#statement: Statement
-instance: ConnectionManager = null
+getConnection(): Connection
+openPreparedStatement(connection: Connection, sql: String): PreparedStatement
+openStatement(connection: Connection): Statement
+closeConnection(Connection connection)
+closeStatement(preparedStatement: PreparedStatement)
+closeStatement(statement: Statement)
+checkConnectionStatus(connection: Connection): boolean
-getInstance(): null
+checkColumnResultValidity(rs: Integer, rs: ResultSet): boolean
+checkMinRow(row: Integer, rs: ResultSet): boolean
+checkResultValidity(row: Integer, col: Integer, rs: ResultSet): boolean
GymPageControlller
-managerModel: ManagerUserModel[1]
-calendarModel: CalendarGymModel[1]
+setModel(calendarModel: CalendarGymModel, managerId: Integer, managerProperty: boolean, gymManager: Gym)
-initializeCalendarGymPage(: MonthPage, dayPage: DayPage, allTrainersList: ObservableList<Trainer>)
+addNewSession(newSession: Session): boolean
-checkTrainerCourse(courseId: Integer, trainer: Trainer)
+setGym(gymId)
CalendarBean
-calendarId: Integer[1]
-calendarproperty: boolean[1]
-currentEntry: Entry<Session>[1]
-gym: Gym[1]
-calendar: Calendar[1]
+getCalendarId(): Integer
+setCalendarId(value: Integer)
+isCalendarId(): boolean
+setCalendarproperty(value: boolean)
+getCalendarCurrentEntry(): Entry<Session>
+setCalendarCurrentEntry(value: Entry<Session>)
+getGym(): Gym
+setGym(value: Gym)
+getCalendar(): Calendar
+setCalendar(value: Calendar)
+update(o: Observable , arg: Object)
CalendarGymModel
-calendarId: Integer[1]
-managerproperty: Boolean[1]
-sessionDAO: SessionDAO[1]
-currentEntry: Entry<Session>[1]
-gym: Gym[1]
-calendar: Calendar[1]
+setCalendarId(calendarId)
+getManagerproperty(): Boolean
+setManagerproperty(value: Boolean)
+getGym(): Gym
+setGym(value: Gym)
+getCalendar(): Calendar
+setCalendar(value: Calendar)
+getCurrentEntry(): Entry<Session>
+setCurrentEntry(value: Entry<Session>)
+getSessionCalendar(): List<Session>
SessionDAO
-instance: SessionDAO = null
+getInstance(): SessionDAO
+insertNewSession(newSession: Session): Integer
+removeSession(sessionToRemove)
CalendarPopupBean
-currentEntry: Entry<Session>
-calendarId: Integer[1]
-sessionGym: Gym[1]
-allTrainersList: ObservableList<Trainer>
+getCurrentEntry(): Entry<Session>
+setCurrentEntry(value: Entry<Session>)
+getCalendarId(): Integer
+setCalendarId(value: Integer)
+getSessionGym(): Gym
+setSessionGym(value: Gym)
+getAllTrainersList(): ObservableList<Trainer>
+setAllTrainersList(value: ObservableList<Trainer>)
+update(o: Observable , arg: Object)
CalendarPopupModel
-currentEntry: Entry<Session>[1]
-calendarId: Integer[1]
-managerProperty: Boolean
-allTrainer: ObservableList<Trainer>
-sessionGym: Gym[1]
+getCurrentEntry(): Entry<Session>
+setCurrentEntry(value: Entry<Session>)
+getCalendarId(): Integer
+setCalendarId(value: Integer)
+isManagerProperty(): Boolean
+setManagerProperty(value: Boolean)
+getAllTrainer(): ObservableList<Trainer>
+setAllTrainer(value: ObservableList<Trainer>)
+getSessionGym(): Gym
+setSessionGym(value: Gym)
+addNewSession(newCalendarEntry: Entry<Session>, currentEntry: Entry<Session>)
CalendarFacade
-calendarGymModel: CalendarGymModel[1]
-monthPage: MonthPage[1]
-dayPage: DayPage[1]
-calendarsEvent: CalendarsEvent[1]
-calendarSource: ObservableList<CalendarSource>[0..*] = FXCollections.observableArrayList()
-allTrainer: ObservableList<Trainer>[1..*]
-gym: Gym[1]
+setCalendar()
-addCalendarSessionList(listSessions: List<Session>)
-fullDayBehaviour()
-leftClickBehaviour()
-fullDayPopOver()
-popOverDialog(param: EntryDetailsPopOverContentParameter): AnchorPane
-addCalendarSessionList(listSession: List<Session>)
CalendarsEvent
-availableCalendar: List<CalendarSource>[1..*]
-availableCalendarsObservableList: ObservableList<CalendarSource>[1..*] = FXCollections.observableArrayList(availableCalendar)
+getCalendarId(): Integer[1]: Calendar
+getCalendarbyName(name: String): Calendar
+getAvailableCalendar(): ObservableList<CalendarSource>
EntryDetailsController
-currentEntry: Entry<Session>[1]
-calendarPopupModel: CalendarPopupModel[1]
-calendarPopupBean: CalendarPopupBean[1]
-courseNameProperty: StringProperty[1]
-timeStartObjectProperty: ObjectProperty[1]
-timeEndObjectProperty: ObjectProperty[1]
-descriptionProperty: StringProperty[1]
-individualCourseBooleanProperty: BooleanProperty[1]
-courseStringProperty: StringProperty[1]
-courseIdIntegerProperty: IntegerProperty[1]
-dateTimeFormatter: DateTimeFormatter[1]
-selectedTrainer: ObjectProperty<MultipleSelectionModel<Trainer>>[1]
+initModel(calendarPopupBean: CalendarPopupBean, calendarPopupModel: CalendarPopupModel, monthPage: MonthPage)
+setRecurrence(event: ActionEvent)
-initNow()
+handle(event: ActionEvent)
+setCourse(event: MouseEvent)

2. ACTIVITY DIAGRAMS

ANDREA EFFICACE

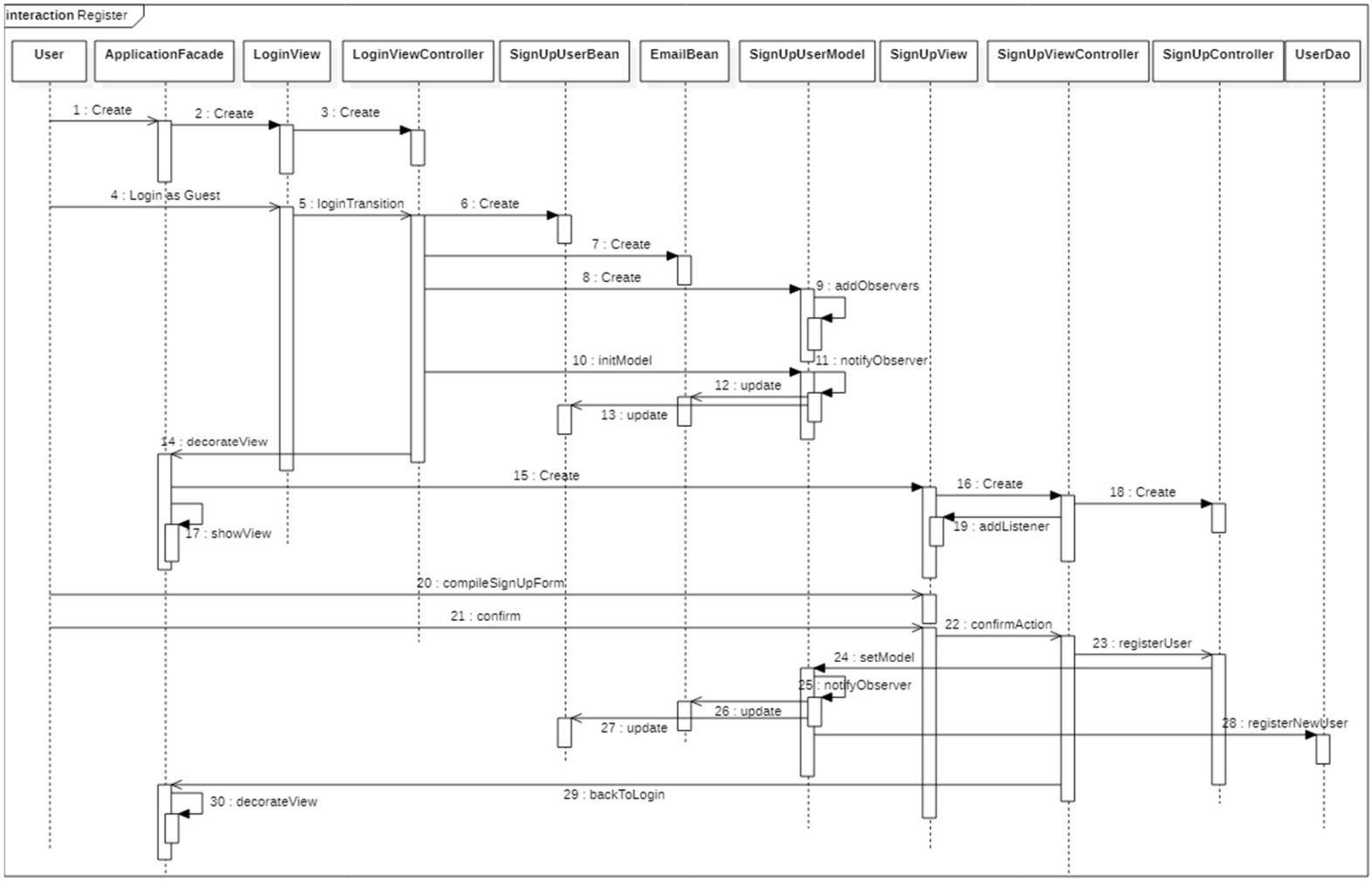


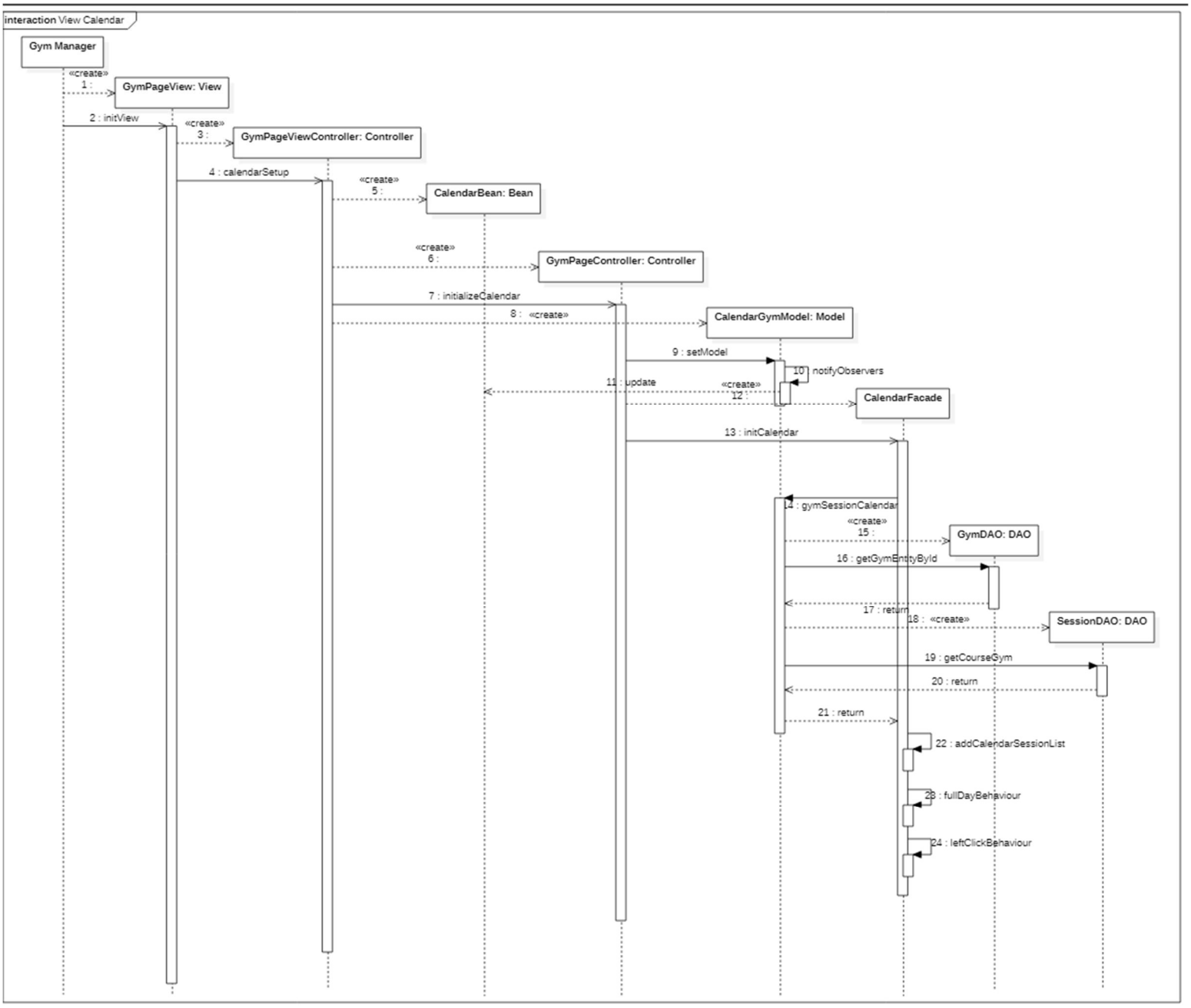

BookingOnMap
Check if user is logged
[isLogged == true]
else
Set user id
request to login
Get user search filters
5 sec
Set search filters
Initialize map*
[isInitialized == true]
Show map
else
Get selected training session
Show timedout connection alert
Show selected training session
[sendEmailBtn == true && bookSessionBtn == false]
[sendEmailBtn == false && bookSessionBtn == false]
Hide selected training session
Show email form
[sendEmailBtn == false && bookSessionBtn == true]
[sendEmailBtn == true && bookSessionBtn == true]
Show "Operation not allowed pop-up
Send email
Book Session*


Initialize map*
Get search filters
Get training session result
Set Google Map key
Connect to GeocodeAPI
Connect to GoogleMapAPI
Geocode results
Set marker list
Set map options
Create map
Add marker to map
Book Session*
Get user id
Get training session Id
Insert on DB new session
else
[isInserted == true]
Show "Session not Booked" alert
Show "Session Booked" pop-up
Load user's booked session

3. SEQUENCE DIAGRAMS

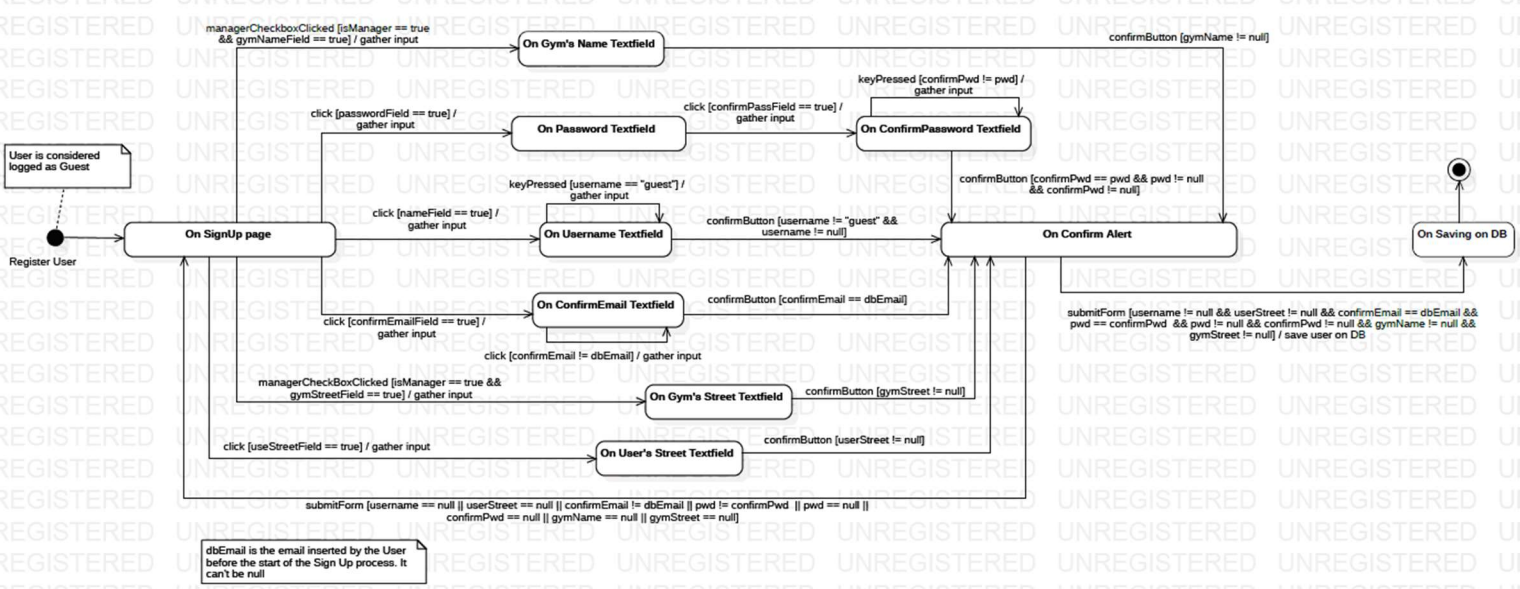
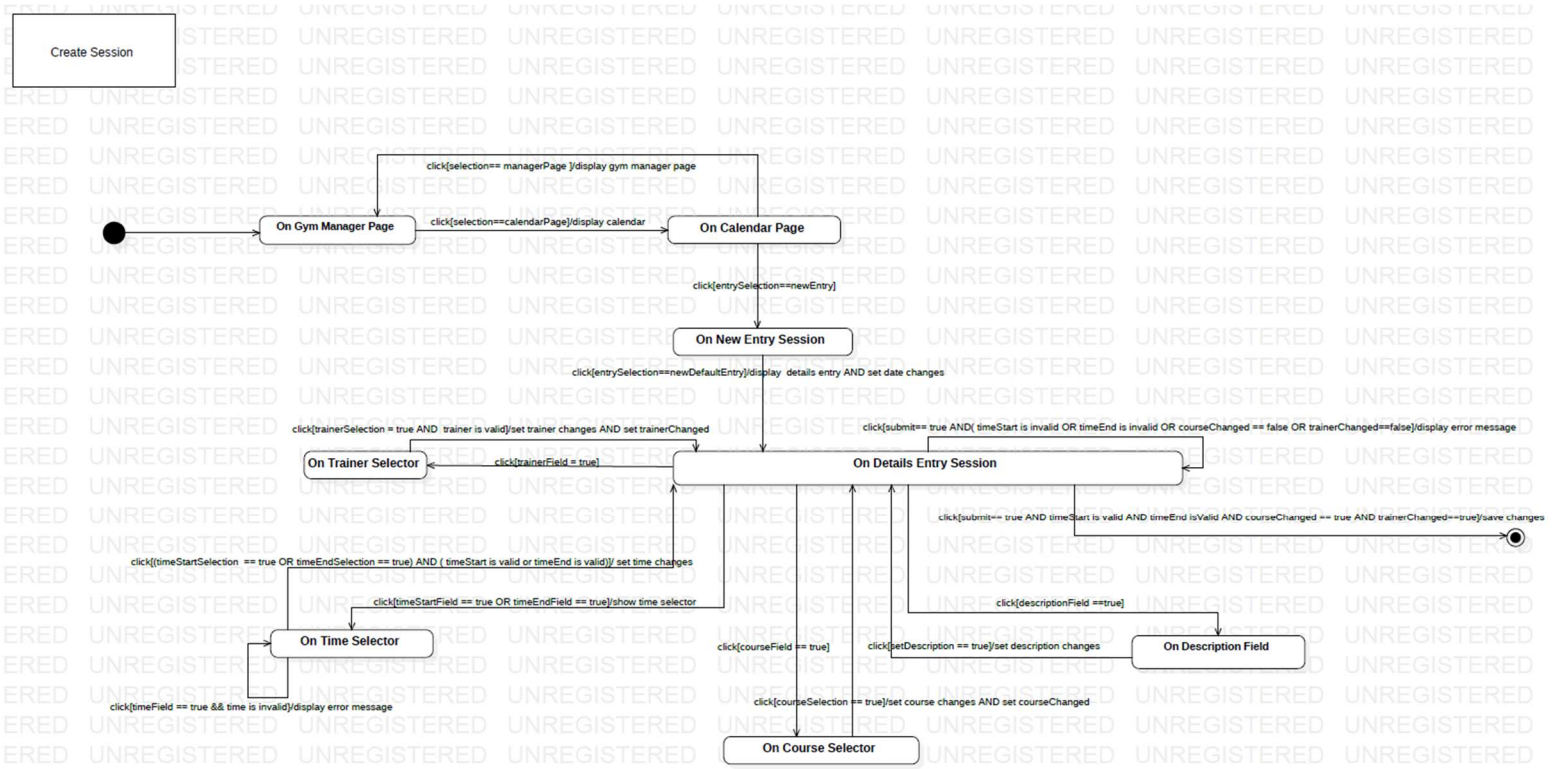
ANDREA EFFICACE




interaction View Calendar
Gym Manager
«create»
1 :
GymPageView: View
2 : initView
«create»
3 :
GymPageViewController: Controller
4 : calendarSetup
«create»
5 :
CalendarBean: Bean
«create»
6 :
GymPageController: Controller
7 : initializeCalendar
8 : «create»
CalendarGymModel: Model
9 : setModel
10 notifyObservers
11 : update
«create»
12 :
CalendarFacade
13 : initCalendar
14 : gymSessionCalendar
«create»
15 :
GymDAO: DAO
16 : getGymEntityById
17 : return
18 : «create»
SessionDAO: DAO
19 : getCourseGym
20 : return
21 : return
22 : addCalendarSessionList
23 : fullDayBehaviour
24 : leftClickBehaviour

4. STATE DIAGRAMS

ANDREA EFFICACE



LORENZO ROSSI