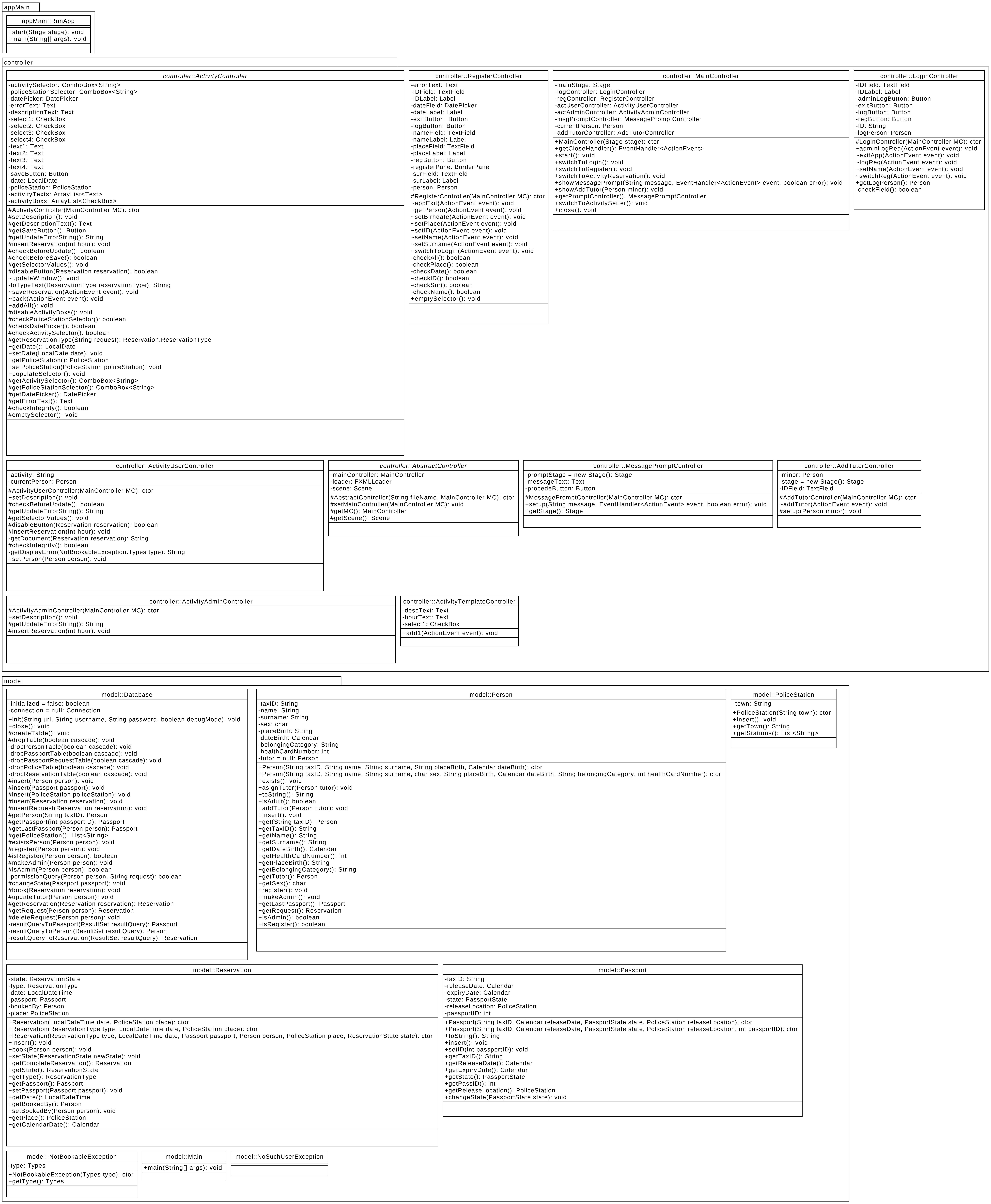

appMain

```
appMain::RunApp

+start(Stage stage): void
+main(String[] args): void
```

controller

```
controller::ActivityController
-activitySelector: ComboBox<String>
-policeStationSelector: ComboBox<String>
-datePicker: DatePicker
-errorText: Text
-descriptionText: Text
-select1: CheckBox
-select2: CheckBox
-select3: CheckBox
-select4: CheckBox
-text1: Text
-text2: Text
-text3: Text
-text4: Text
-saveButton: Button
-date: LocalDate
-policeStation: PoliceStation
-activityTexts: ArrayList<Text>
-activityBoxs: ArrayList<CheckBox>
#ActivityController(MainController MC): ctor
#setDescription(): void
#getDescriptionText(): Text
#getSaveButton(): Button
#getUpdateErrorString(): String
#insertReservation(int hour): void
#checkBeforeUpdate(): boolean
#checkBeforeSave(): boolean
#getSelectorValues(): void
#disableButton(Reservation reservation): boolean
~updateWindow(): void
-toTypeText(ReservationType reservationType): String
~saveReservation(ActionEvent event): void
~back(ActionEvent event): void
+addAll(): void
#disableActivityBoxs(): void
#checkPoliceStationSelector(): boolean
#checkDatePicker(): boolean
#checkActivitySelector(): boolean
#getReservationType(String request): Reservation.ReservationType
+getDate(): LocalDate
+setDate(LocalDate date): void
+getPoliceStation(): PoliceStation
+setPoliceStation(PoliceStation policeStation): void
+populateSelector(): void
#getActivitySelector(): ComboBox<String>
#getPoliceStationSelector(): ComboBox<String>
#getDatePicker(): DatePicker
#getErrorText(): Text
#checkIntegrity(): boolean
#emptySelector(): void
```

```
controller::RegisterController
-errorText: Text
-IDField: TextField
-IDLabel: Label
-dateField: DatePicker
-dateLabel: Label
-exitButton: Button
-logButton: Button
-nameField: TextField
-nameLabel: Label
-placeField: TextField
-placeLabel: Label
-regButton: Button
-registerPane: BorderPane
-surField: TextField
-surLabel: Label
-person: Person
#RegisterController(MainController MC): ctor
~appExit(ActionEvent event): void
~getPerson(ActionEvent event): void
~setBirhdate(ActionEvent event): void
~setPlace(ActionEvent event): void
~setID(ActionEvent event): void
~setName(ActionEvent event): void
~setSurname(ActionEvent event): void
~switchToLogin(ActionEvent event): void
-checkAll(): boolean
-checkPlace(): boolean
-checkDate(): boolean
-checkID(): boolean
-checkSur(): boolean
-checkName(): boolean
+emptySelector(): void
```

```
controller::MainController
-mainStage: Stage
-logController: LoginController
-regController: RegisterController
-actUserController: ActivityUserController
-actAdminController: ActivityAdminController
-msgPromptController: MessagePromptController
-currentPerson: Person
-addTutorController: AddTutorController
+MainController(Stage stage): ctor
+getCloseHandler(): EventHandler<ActionEvent>
+start(): void
+switchToLogin(): void
+switchToRegister(): void
+switchToActivityReservation(): void
+showMessagePrompt(String message, EventHandler<ActionEvent> event, boolean error): void
+showAddTutor(Person minor): void
+getPromptController(): MessagePromptController
+switchToActivitySetter(): void
+close(): void
```

```
controller::LoginController
-IDField: TextField
-IDLabel: Label
-adminLogButton: Button
-exitButton: Button
-logButton: Button
-regButton: Button
-ID: String
-logPerson: Person
#LoginController(MainController MC): ctor
~adminLogReq(ActionEvent event): void
~exitApp(ActionEvent event): void
~logReq(ActionEvent event): void
~setName(ActionEvent event): void
~switchReg(ActionEvent event): void
+getLogPerson(): Person
-checkField(): boolean
```

```
controller::ActivityUserController
-activity: String
-currentPerson: Person
#ActivityUserController(MainController MC): ctor
+setDescription(): void
#checkBeforeUpdate(): boolean
#getUpdateErrorString(): String
#getSelectorValues(): void
#disableButton(Reservation reservation): boolean
#insertReservation(int hour): void
-getDocument(Reservation reservation): String
#checkIntegrity(): boolean
-getDisplayError(NotBookableException.Types type): String
+setPerson(Person person): void
```

```
controller::AbstractController
-mainController: MainController
-loader: FXMLLoader
-scene: Scene
#AbstractController(String fileName, MainController MC): ctor
#setMainController(MainController MC): void
#getMC(): MainController
#getScene(): Scene
```

```
controller::MessagePromptController
-promptStage = new Stage(): Stage
-messageText: Text
-procedeButton: Button
#MessagePromptController(MainController MC): ctor
+setup(String message, EventHandler<ActionEvent> event, boolean error): void
+getStage(): Stage
```

```
controller::AddTutorController
-minor: Person
-stage = new Stage(): Stage
-IDField: TextField
#AddTutorController(MainController MC): ctor
~addTutor(ActionEvent event): void
#setup(Person minor): void
```

```
controller::ActivityAdminController
#ActivityAdminController(MainController MC): ctor
+setDescription(): void
#getUpdateErrorString(): String
#insertReservation(int hour): void
```

```
controller::ActivityTemplateController
-descText: Text
-hourText: Text
-select1: CheckBox
~add1(ActionEvent event): void
```

model

```
model::Database
-initialized = false: boolean
-connection = null: Connection
+init(String url, String username, String password, boolean debugMode): void
+close(): void
#createTable(): void
#dropTable(boolean cascade): void
-dropPersonTable(boolean cascade): void
-dropPassportTable(boolean cascade): void
-dropPassportRequestTable(boolean cascade): void
-dropPoliceTable(boolean cascade): void
-dropReservationTable(boolean cascade): void
#insert(Person person): void
#insert(Passport passport): void
#insert(PoliceStation policeStation): void
#insert(Reservation reservation): void
#insertRequest(Reservation reservation): void
#getPerson(String taxID): Person
#getPassport(int passportID): Passport
#getLastPassport(Person person): Passport
#getPoliceStation(): List<String>
#existsPerson(Person person): void
#register(Person person): void
#isRegister(Person person): boolean
#makeAdmin(Person person): void
#isAdmin(Person person): boolean
-permissionQuery(Person person, String request): boolean
#changeState(Passport passport): void
#book(Reservation reservation): void
#updateTutor(Person person): void
#getReservation(Reservation reservation): Reservation
#getRequest(Person person): Reservation
#deleteRequest(Person person): void
-resultQueryToPassport(ResultSet resultQuery): Passport
-resultQueryToPerson(ResultSet resultQuery): Person
-resultQueryToReservation(ResultSet resultQuery): Reservation
```

```
model::Person
-taxID: String
-name: String
-surname: String
-sex: char
-placeBirth: String
-dateBirth: Calendar
-belongingCategory: String
-healthCardNumber: int
-tutor = null: Person
+Person(String taxID, String name, String surname, String placeBirth, Calendar dateBirth): ctor
+Person(String taxID, String name, String surname, char sex, String placeBirth, Calendar dateBirth, String belongingCategory, int healthCardNumber): ctor
+exists(): void
+asignTutor(Person tutor): void
+toString(): String
+isAdult(): boolean
+addTutor(Person tutor): void
+insert(): void
+get(String taxID): Person
+getTaxID(): String
+getName(): String
+getSurname(): String
+getDateBirth(): Calendar
+getHealthCardNumber(): int
+getPlaceBirth(): String
+getBelongingCategory(): String
+getTutor(): Person
+getSex(): char
+register(): void
+makeAdmin(): void
+getLastPassport(): Passport
+getRequest(): Reservation
+isAdmin(): boolean
+isRegister(): boolean
```

```
model::PoliceStation
-town: String
+PoliceStation(String town): ctor
+insert(): void
+getTown(): String
+getStations(): List<String>
```

```
model::Reservation
-state: ReservationState
-type: ReservationType
-date: LocalDateTime
-passport: Passport
-bookedBy: Person
-place: PoliceStation
+Reservation(LocalDateTime date, PoliceStation place): ctor
+Reservation(ReservationType type, LocalDateTime date, PoliceStation place): ctor
+Reservation(ReservationType type, LocalDateTime date, Passport passport, Person person, PoliceStation place, ReservationState state): ctor
+insert(): void
+book(Person person): void
+setState(ReservationState newState): void
+getCompleteReservation(): Reservation
+getState(): ReservationState
+getType(): ReservationType
+getPassport(): Passport
+setPassport(Passport passport): void
+getDate(): LocalDateTime
+getBookedBy(): Person
+setBookedBy(Person person): void
+getPlace(): PoliceStation
+getCalendarDate(): Calendar
```

```
model::Passport
-taxID: String
-releaseDate: Calendar
-expiryDate: Calendar
-state: PassportState
-releaseLocation: PoliceStation
-passportID: int
+Passport(String taxID, Calendar releaseDate, PassportState state, PoliceStation releaseLocation): ctor
+Passport(String taxID, Calendar releaseDate, PassportState state, PoliceStation releaseLocation, int passportID): ctor
+toString(): String
+insert(): void
+setID(int passportID): void
+getTaxID(): String
+getReleaseDate(): Calendar
+getExpiryDate(): Calendar
+getState(): PassportState
+getPassID(): int
+getReleaseLocation(): PoliceStation
+changeState(PassportState state): void
```

```
model::NotBookableException
-type: Types
+NotBookableException(Types type): ctor
+getType(): Types
```

```
model::Main

+main(String[] args): void
```

```
model::NoSuchUserException
```