TortoiseHospital

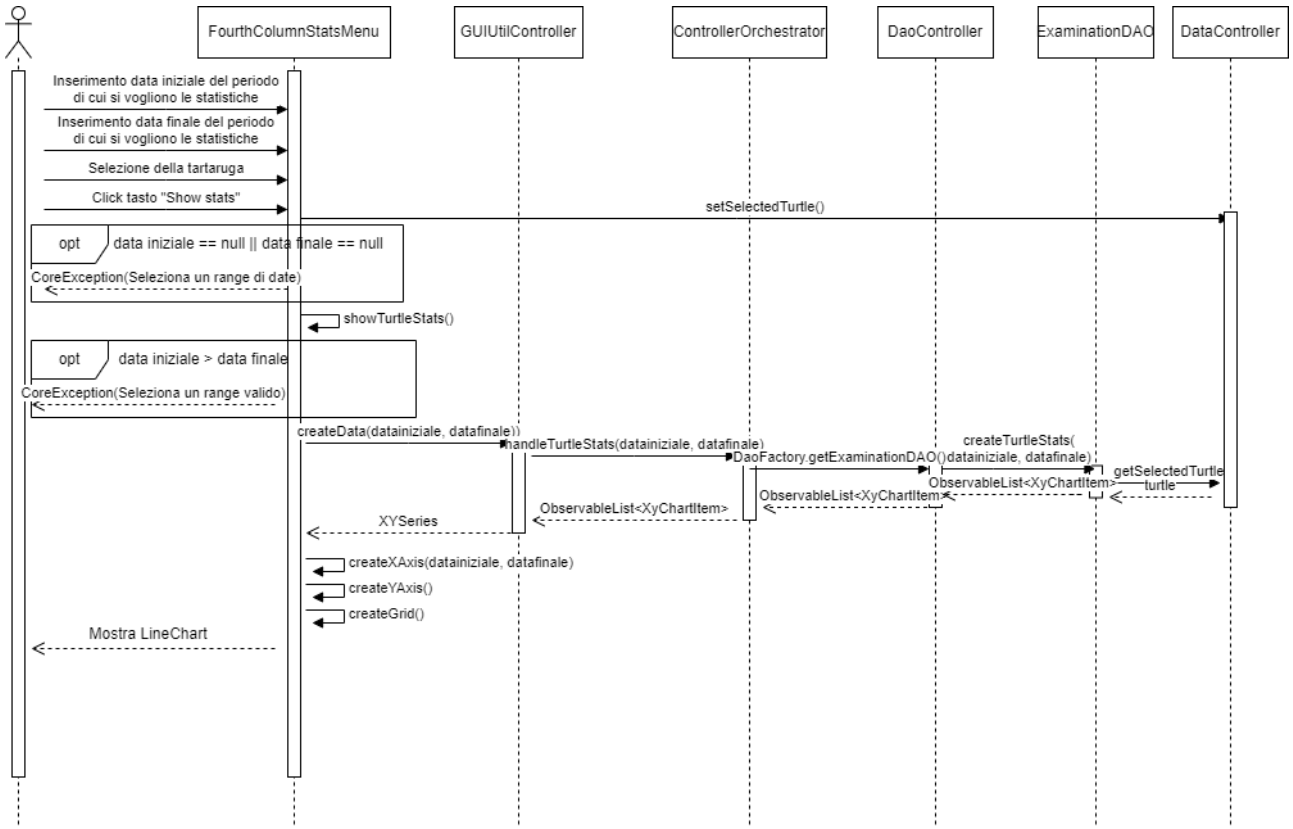
PROGETTAZIONE E SVILUPPO DI UNA DESKTOP APP PER LA GESTIONE DEI RICOVERI DI TARTARUGHE MARINE NEI CENTRI DI RECUPERO NAZIONALI

Eduardo Gaudiosi

N86003812

Pagina lasciata volutamente bianca.

INDICE



1. DESCRIZIONE DEL PROGETTO

Il progetto si concentra sulla creazione di un sistema informativo per la gestione dei centri di recupero nazionali per tartarughe marine. Questo sistema comprenderà una base di dati relazionale e un'applicazione Java con un'interfaccia utente grafica (GUI) per semplificare la gestione delle tartarughe e dei dati dei centri. Sarà possibile registrare l'ingresso e la riammissione delle tartarughe, assegnando loro identificativi univoci e aggiornando le informazioni sulla loro salute. Il sistema offrirà inoltre la possibilità di visualizzare storici dettagliati delle tartarughe, nonché statistiche mensili e annuali per monitorare il numero di tartarughe accolte e il loro stato di salute. Inoltre, consentirà la gestione dei dati dei centri e del personale addetto.

Questo progetto risponde alla necessità di migliorare la gestione e la tracciabilità delle tartarughe marine presso i centri di recupero, contribuendo alla loro conservazione e alla ricerca scientifica. L'utilizzo di una base di dati relazionale e un'interfaccia utente intuitiva renderà più efficienti le operazioni quotidiane nei centri, consentendo una registrazione accurata dei dati e una valutazione completa dello stato di salute delle tartarughe. Inoltre, le statistiche forniranno una panoramica utile per la pianificazione e l'analisi delle attività di recupero. Il progetto consiste nello sviluppo di un sistema informativo completo per la gestione del ricovero di tartarughe marine presso i centri di Recupero Nazionali. Il sistema comprenderà una base di dati relazionale per la memorizzazione dei dati fondamentali dei centri e delle tartarughe, nonché un'applicazione Java con interfaccia grafica utente (GUI), basata su JavaFX, per la gestione delle operazioni di registrazione, accesso, riammissione e monitoraggio delle tartarughe marine.

2. OVERVIEW DELL'ARCHITETTURA

Considerata la traccia, e soprattutto dovendo gestire potenzialmente una moltitudine di centri, si è preferito seguire una struttura three-tier. Vantaggio principale dell'architettura è che le componenti possano essere scalate indipendentemente l'una dall'altra e una flessibilità tale da non impattare gli altri tier nel caso di update del presentation layer (in particolare, delle librerie di cui il presentation layer fa uso, prime tra tutte ControlsFX, MaterialFX e Charts).

A supporto dell'architettura three-tier, nel Data tier è compreso l'utilizzo di classi DAO (sotto il package me.csaprotocol.tortoisehospital.daos) e di classi DTO per gestire lo storage degli oggetti (package me.csaprotocol.tortoisehospital.entities).

Al fine di evitare interrogazioni ad albero estremamente costose al database e duplicazioni di dati non realmente necessari, si è deciso di non memorizzare entità in altre entità, optando per un sistema di caching all'interno della classe DataController, creata tramite il singleton pattern e di cui dunque può esistere una sola istanza: inoltre, DataController è modificato ad ogni interrogazione, assicurando che i dati all'interno siano sempre coerenti con quelli all'interno del database.

Infine, presupponendo un update futuro in cui saranno supportati diversi tipi di database, si è già provveduto a creare la struttura composta di interfacce, classi astratte e DAOFactory che permetterà di sviluppare il supporto per altri database senza andare a modificare le chiamate a metodi di DAOController. Sarà inoltre possibile selezionare il database per l'utente semplicemente modificando il file db.properties (attualmente inserito direttamente nel jar per l'esistenza di una sola e singola opzione).

3. DIPENDENZE

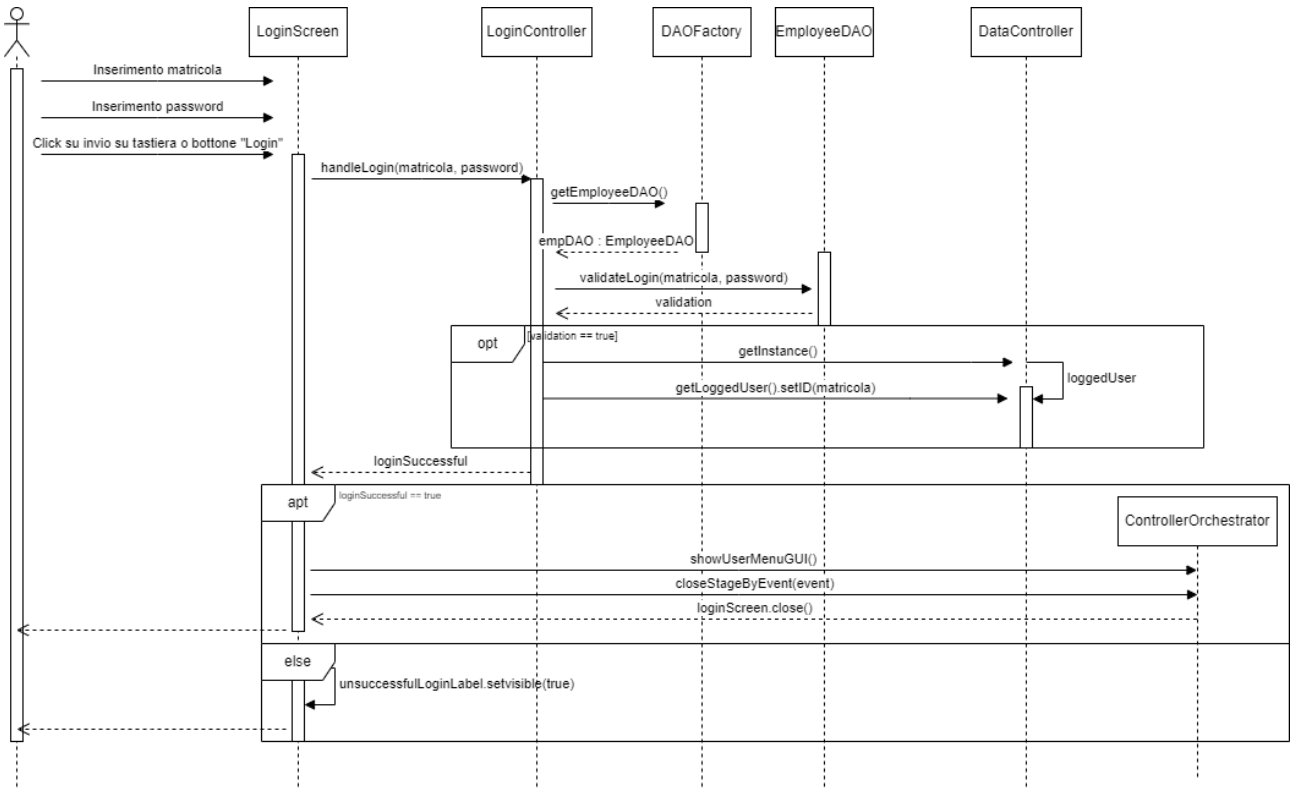
Il progetto utilizza varie librerie di supporto per JavaFX al fine di integrare GUI non incluse nel pacchetto originale. Inoltre, sono implementate librerie per ottimizzare la leggibilità e la compattezza del codice. Per la gestione di queste librerie, si è fatto ricorso a Maven.

1. **ControlsFX [org.controlsfx.controlsfx]** : Si tratta di una libreria che possiede diversi tipi di strumenti di supporto per JavaFX, nonché una delle più popolari nell'ambito del linguaggio. In TortoiseHospital, è stata utilizzata sia per creare delle notifiche, sia per creare i PopOver che spiegano le funzionalità della GUI a runtime, rendendola più intuitiva.

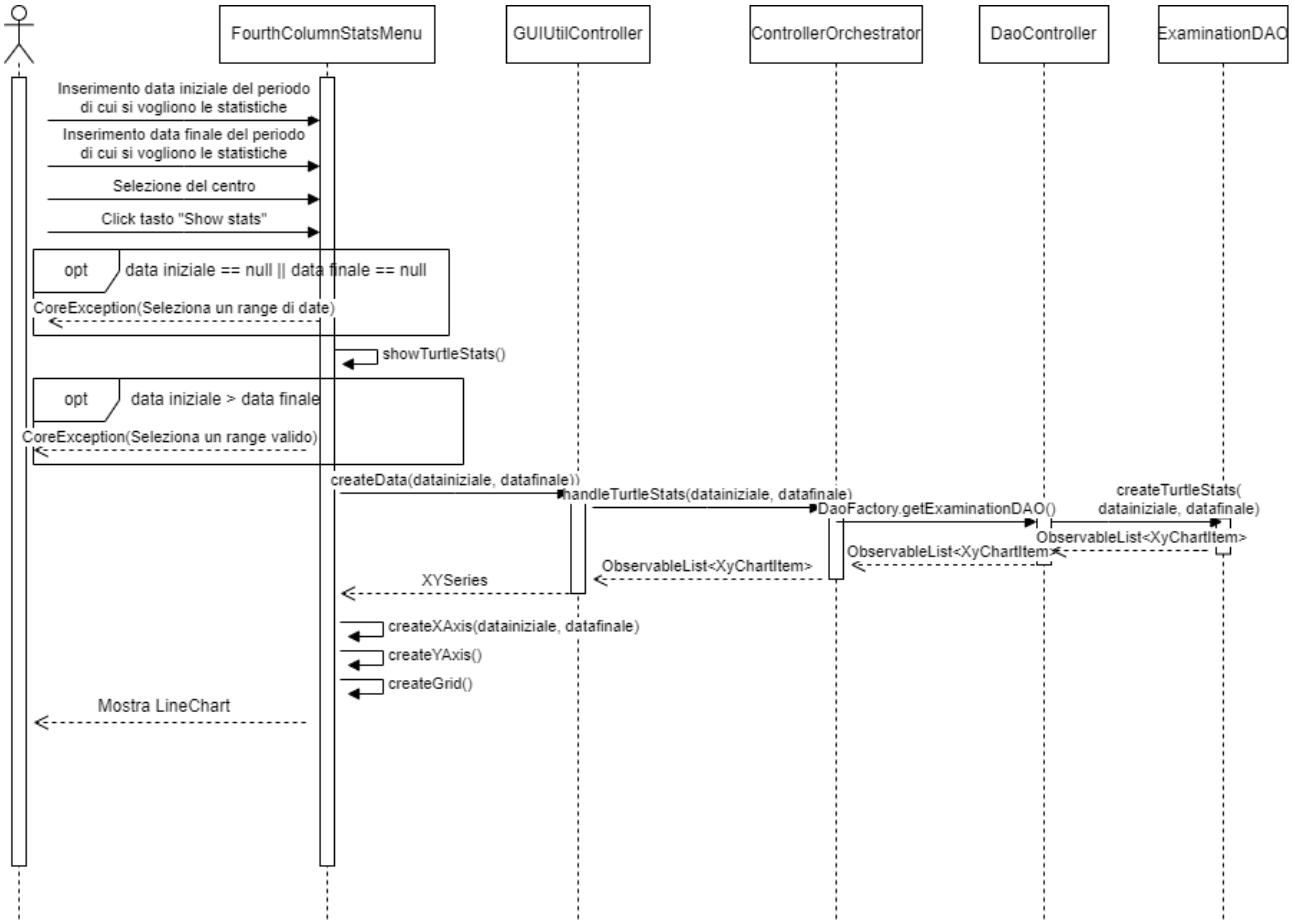
2. **HikariCP [com.zaxxer.hikaricp]** : Libreria di supporto per il connection pooling. Si è scelto l'utilizzo di hikari dopo aver consultato i benchmark di Hikari confrontati a quelli di diverse altre librerie che offrono le stesse funzioni (consultabili qui).
3. **Project Lombok [org.projectlombok.lombok]** : Libreria che rende il codice molto più leggibile attraverso l'uso di annotations. Nonostante esista in Java la possibilità di creare record, questi ultimi non risultano modificabili. Si è deciso dunque , per necessità di modificare le entities (nello specifico all'interno del DataController), di usare Lombok invece dei record.
4. **Google Guava [com.google.guava.guava]** : Di questa libreria è stato utilizzato in maniera molto consistente l'EventBus, che dà la possibilità di inviare Events di JavaFX a diverse delle strutture della GUI allo stesso tempo. Sarebbe stato impossibile gestire questo tipo di comunicazione tra GUI usando gli Events di JavaFX poiché, una volta che l'evento è stato lanciato, viene consumato da un singolo listener impedendo ad altre strutture di recepirlo a loro volta.
5. **MaterialFX [io.github.palexdev.materialfx]** : MaterialFX è la libreria che in assoluto è stata utilizzata di più all'interno del codice. Le varie estensioni di Button, TextField eccetera sono costruite seguendo i principi di material design di Google, il che rende le GUI create a partire da questi elementi piacevoli alla vista e soprattutto con un design molto moderno.

4. SEQUENCE DIAGRAM

Il seguente sequence diagram mostra la funzionalità di login present all'interno dell'app. Per design, ad ogni singolo utente vengono mostrati solo ed esclusivamente i centri in cui è autorizzato a lavorare: da qui, l'utilità di avere una variabile loggedUser all'interno di DataController.



Il seguente sequence diagram mostra invece l'iter che l'applicazione segue per poter generare le statistiche di una specifica tartaruga. Si è preferita, alla libreria di default per gestire i grafici, usare Charts dello sviluppatore HanSolo, una libreria contenente molti tipi di grafici usabili a scopo scientifico.



5. CLASS DIAGRAM

Il seguente class diagram mostra le varie classi presenti nel progetto e le relazioni tra di esse. Nota: all'interno di questo file si è preferito inserire un link per le limitazioni imposte dall'export di draw.io, app utilizzata per crearlo. Infatti, esportando in formato SVG, si potrebbero avere dei problemi di visualizzazione. Dunque, si è preferito creare un pdf sia consultabile che scaricabile al fine di porre rimedio a tali eventuali problemi di visualizzazione.

Click qui per visualizzare il class diagram

6. CRC CARDS

ControllerOrchestrator me.csaprotocol.tortoisehospital.controllers	
ControllerOrchestrator si occupa di mostrare GUI di diverso tipo all'utente e di supervisionare l'interagire tra le GUI facendo da intermediario.	Interagisce con tutte le entità e con gli fxmlcontroller della terza e quarta colonna del turtleMenu.

DaoController me.csaprotocol.tortoisehospital.controllers	
DaoController si occupa di monitorare l'accesso al data layer e alla classe DAOFactory.	Interagisce con la classe DAOFactory.

DataController me.csaprotocol.tortoisehospital.controllers	
DataController si occupa di servire da cache per l'app. Si tratta di una classe definita con il pattern singleton.	Non interagisce con nessun'altra classe.

EventController me.csaprotocol.tortoisehospital.controllers	
EventController si occupa di lanciare gli eventi all'interno di eventbus.	Interagisce con l'intero package events.

GUIUtilsController me.csaprotocol.tortoisehospital.controllers	
GuiUtilsController si occupa di fornire all'interfaccia nuovi nodi, se dovessero servirne.	Interagisce con gli fxmlcontroller dei vari bottoni.

LoginController me.csaprotocol.tortoisehospital.controllers	
LoginController si occupa di gestire la funzione login dell'app.	Interagisce con ControllerOrchestrator

DAOFactory me.csaprotocol.tortoisehospital.daos.factory	
DaoFactory è la classe che gestisce i DAO. Generando DAO attraverso questa classe, è possibile supportare diversi tipi di database.	Interagisce con l'intero package daos.

ConnectionPoolPostgres me.csaprotocol.tortoisehospital.daos.pgsql.jdbc	
ConnectionPoolPostgres gestisce la connection pool dell'applicazione.	Classe standalone, non interagisce con altre classi del sistema.

<<abstract class>> PostgresDAO me.csaprotocol.tortoisehospital.daos.pgsql.jdbc	
Classe astratta che fa storage della dataSource. Implementata per facilitare un'eventuale transizione da una libreria ad una differente.	Interagisce con ConnectionPoolPostgres.

CenterDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Center del database.	Interagisce con l'entità Center.

EmployeeDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Employee del database.	Interagisce con l'entità Employee.

ExaminationDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Examination del database.	Interagisce con l'entità Examination.

MeasurementDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Measurement del database.	Interagisce con l'entità Measurement.

MedicalRecordDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Medical Record del database.	Interagisce con l'entità MedicalRecord.

TankDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Tank del database.	Interagisce con l'entità Tank.

TurtleDAOPostgres me.csaprotocol.tortoisehospital.daos.pgsql	
Si occupa di gestire tutte le interrogazioni su tabella Turtle del database.	Interagisce con l'entità Turtle.

<<interface>> CenterDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi CenterDAO	Collabora con l'entità Center.

<<interface>> EmployeeDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi EmployeeDAO	Collabora con l'entità Employee.

<<interface>> ExaminationDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi ExaminationDAO.	Collabora con l'entità Examination.

<<interface>> MeasurementDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi MeasurementDAO	Collabora con l'entità Measurement.

<<interface>> MedicalRecordDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi MedicalRecordDAO	Collabora con l'entità MedicalRecord.

<<interface>> TankDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi TankDAO	Collabora con l'entità Tank.

<<interface>> CenterDAO me.csaprotocol.tortoisehospital.daos	
Interfaccia implementata da tutte le classi TurtleDAO	Collabora con l'entità Turtle.

Center me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella Center dello schema concettuale del database.	Non interagisce con altre classi.

Employee me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella Employee dello schema concettuale del database.	Non interagisce con altre classi.

Examination me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella Examination dello schema concettuale del database.	Dipende dall'enumerazione Status.

Status me.csaprotocol.tortoisehospital.entities.enums	
Definisce un tipo per lo Status delle parti di una tartaruga, i colori e i valori a loro corrispondenti.	Non interagisce con altre classi.

Measurement me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella measurement dello schema concettuale del database.	Non interagisce con altre classi.

Medical Record me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella medical_record dello schema concettuale del database.	Non interagisce con altre classi.

Tank me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella Tank dello schema concettuale del database.	Non interagisce con altre classi.

Turtle me.csaprotocol.tortoisehospital.entities	
Entità corrispondente alla tabella Turtle dello schema concettuale del database.	Dipende dall'enumerazione Sex per definire il sesso di una tartaruga.

Sex me.csaprotocol.tortoisehospital.entities.enums	
Definisce un tipo per il sesso delle tartarughe.	Non interagisce con altre classi.

EventBus me.csaprotocol.tortoisehospital.events	
Classe creata con pattern singleton, si occupa di far sopraggiungere gli eventi lanciati a tutti nodi della gui "iscritti"	Non interagisce con altre classi.

CenterClickEvent me.csaprotocol.tortoisehospital.events	
Classe che estende Events di JavaFX. L'evento comunica ai nodi "iscritti" che un centro è stato cliccato.	Non interagisce con altre classi.

ExaminationClickEvent me.csaprotocol.tortoisehospital.events	
Classe che estende Events di JavaFX. L'evento comunica ai nodi "iscritti" che un esame è stato cliccato.	Non interagisce con altre classi.

MeasurementClickEvent me.csaprotocol.tortoisehospital.events	
Classe che estende Events di JavaFX. L'evento comunica ai nodi "iscritti" che una misurazione è stata cliccata.	Non interagisce con altre classi.

MedicalRecordClickEvent me.csaprotocol.tortoisehospital.events	
Classe che estende Events di JavaFX. L'evento comunica ai nodi "iscritti" che una cartella medica è stata cliccata.	Non interagisce con altre classi.

TankClickEvent me.csaprotocol.tortoisehospital.events	
Classe che estende Events di JavaFX. L'evento comunica ai nodi "iscritti" che una vasca è stata cliccata.	Non interagisce con altre classi.

CoreException me.csaprotocol.tortoisehospital.exceptions	
Estende Exception. Viene usata per segnalare errori nei tier logic/presentation.	Non interagisce con altre classi.

DAOException me.csaprotocol.tortoisehospital.exceptions	
Estende Exception. Viene usata per segnalare errori nel data tier.	Non interagisce con altre classi.

ExceptionHandler me.csaprotocol.tortoisehospital.exceptions	
Handler custom per exception. Crea delle notifiche grafiche per segnalare errori all'utente ed esortarlo a chiamare un amministratore di sistema.	Non interagisce con altre classi.

fourthColumnStatsMenu me.csaprotocol.tortoisehospital.fxmlcontrollers.modularmenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per visualizzare le statistiche di una tartaruga.	Interagisce con GUIUtilsController, CoreException ed ExceptionHandler.

thirdColumnStatsMenu me.csaprotocol.tortoisehospital.fxmlcontroller.modularmenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per visualizzare le statistiche di un centro.	Interagisce con ControllerOrchestrator, GUIUtilsController, CoreException ed ExceptionHandler.

thirdColumnTurtleMenu me.csaprotocol.tortoisehospital.fxmlcontrollers.modularmenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per visualizzare i dettagli di una tartaruga e le sue misurazioni.	Interagisce con ControllerOrchestrator e GUIUtilsController.

fourthColumnStatsMenu me.csaprotocol.tortoisehospital.fxmlcontrollers.modularmenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per visualizzare le cartelle mediche e gli esami associati ad una tartaruga.	Interagisce con ControllerOrchestrator e GUIUtilsController.

userMenu me.csaprotocol.tortoisehospital.fxmlcontrollers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia principale per la gestione dell'ospedale.	Interagisce con ControllerOrchestrator e GUIUtilsController.

LoginScreen me.csaprotocol.tortoisehospital.fxmlcontrollers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per il login degli impiegati.	Interagisce con ControllerOrchestrator, GUIUtilsController, CoreException ed ExceptionHandler.

DialogUtil me.csaprotocol.tortoisehospital.fxmlcontrollers	
Classe di supporto per la creazione di finestre di dialogo. Viene usata principalmente dalle classi nel package steppers.	Interagisce con ControllerOrchestrator.

NewExamination me.csaprotocol.tortoisehospital. fxmlcontrollers.steppers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per la creazione di un nuovo esame.	Interagisce con ControllerOrchestrator e GUIUtilsController.

NewMedicalRecord me.csaprotocol.tortoisehospital. fxmlcontrollers.steppers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per la creazione di una nuova cartella medica.	Interagisce con ControllerOrchestrator e GUIUtilsController.

NewMeasurement me.csaprotocol.tortoisehospital. fxmlcontrollers.steppers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per la creazione di una nuova misurazione.	Interagisce con ControllerOrchestrator e GUIUtilsController.

NewTurtle me.csaprotocol.tortoisehospital.fxmlcontrollers.steppers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per la creazione di una nuova tartaruga.	Interagisce con ControllerOrchestrator e GUIUtilsController.

UpdateExamination me.csaprotocol.tortoisehospital.fxmlcontrollers.steppers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per l'aggiornamento di un esame.	Interagisce con ControllerOrchestrator , GUIUtilsController e DataController.

UpdateTurtle me.csaprotocol.tortoisehospital.fxmlcontrollers.steppers	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per l'aggiornamento delle specifiche di una tartaruga.	Interagisce con ControllerOrchestrator , GUIUtilsController e DataController.

CenterButton me.csaprotocol.tortoisehospital.fxmlcontrollers.usermenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per bottone cliccabile dedicato ad un singolo centro.	Interagisce con ControllerOrchestrator ed EventController.

ExaminationButton me.csaprotocol.tortoisehospital.fxmlcontrollers.usermenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per bottone cliccabile dedicato ad un singolo esame.	Interagisce con ControllerOrchestrator ed EventController.

MeasurementButton me.csaprotocol.tortoisehospital.fxmlcontrollers.usermenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per bottone cliccabile dedicato ad una singola misurazione.	Interagisce con ControllerOrchestrator ed EventController.

MedicalRecordButton me.csaprotocol.tortoisehospital.fxmlcontrollers.usermenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per bottone cliccabile dedicato ad una singola cartella medica.	Interagisce con ControllerOrchestrator ed EventController.

TankButton me.csaprotocol.tortoisehospital.fxmlcontrollers.usermenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per bottone cliccabile dedicato ad una singola vasca.	Interagisce con ControllerOrchestrator ed EventController.

TurtleButton me.csaprotocol.tortoisehospital.fxmlcontrollers.usermenu	
Controller associato all'omonimo file FXML. Provvede al fornire i metodi chiamati dal file FXML. Interfaccia per bottone cliccabile dedicato ad una singola tartaruga.	Interagisce con ControllerOrchestrator ed EventController.