Documentation

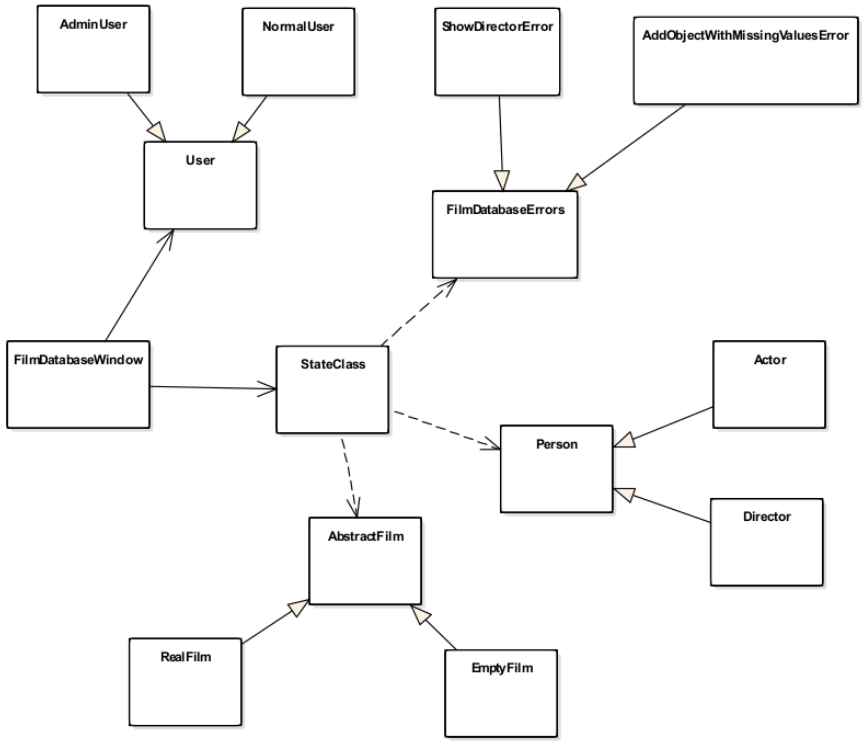
Film collector app – a movie database

1) Installation and Start up

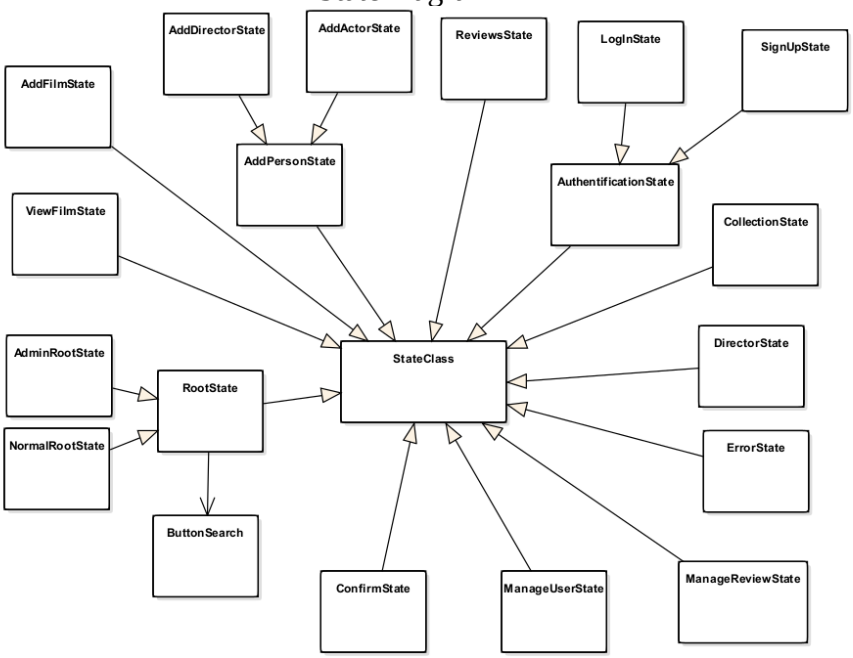
- download and install mongo database to your computer
  (https://docs.mongodb.com/manual/administration/install-community/)
- start mongo database
- download Pharo
- in Pharo tools/catalogBrowser install VoyageMongo
- File in provided package (Semestralka.st) into your Pharo image
- start app with examples data with FilmDatabaseWindow exampleCleanAndInit
- return to existing DB from playground with FilmDatabaseWindow exampleDatabaseReturn

2) Architecture of our solution

AppDiagram

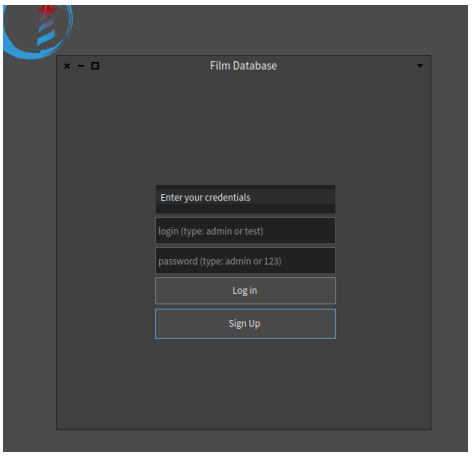


StateDiagram

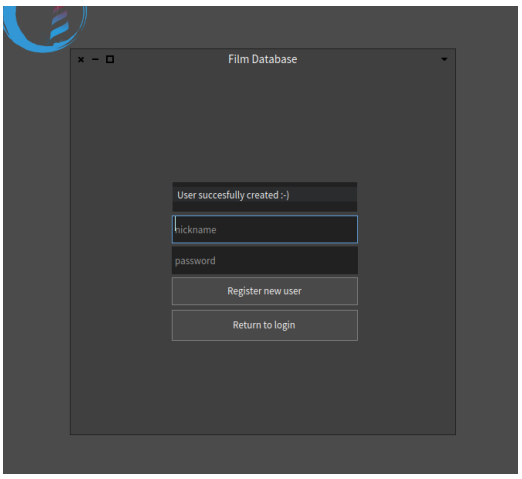


3) Use case – examples of our implemented features

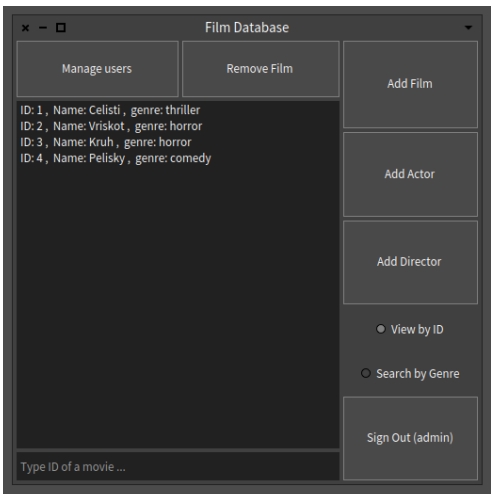
LogIn



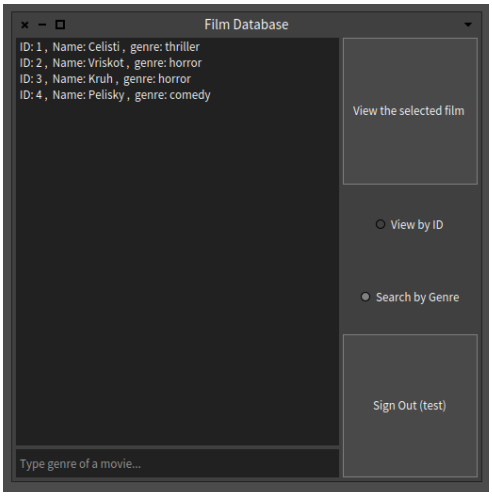
SignUp



MainAdminView

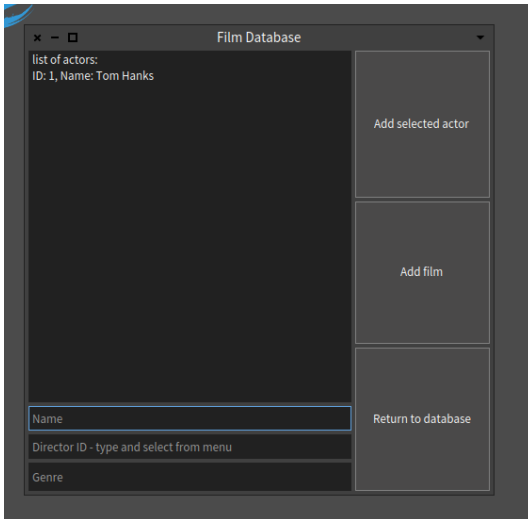


MainUserView



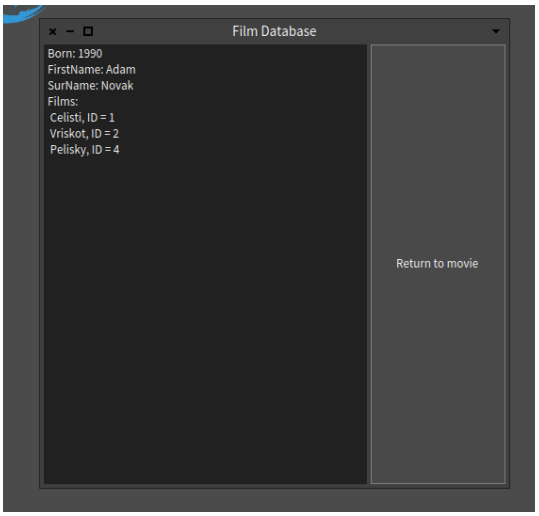
ConfirmDelete

AddingFilm

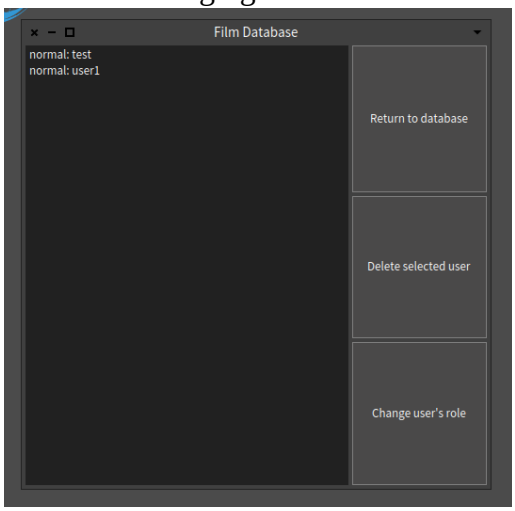


ShowFilmInfo

ShowDirectorInfo



ManagingUsers



ReviewsOfFilm

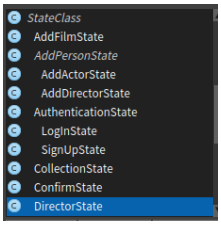
4) Architectural decisions

- we are using Pharo as pure object oriented language for its easy syntax among others
- we are using Mongo database for easy way how to store data + in Pharo there is already implemented framework for communication with Mongo database
- we are using State pattern for easy implementation and simple adding of new states (which in our case is implementing new features)
- we are using Spec UI framework as graphical interface. It‘s very easy to use and thanks to great documentation very easy to learn

5) Design patterns

➔ **Polymorfism**

- StateClass




```
initializePresenter
    buttonReturn action: [ myWindow dynamicChange: ViewMovieState ].
```

```
dynamicChange: aStateClass
    self state: aStateClass.
    self needRebuild: false.
    self buildWithSpecLayout: self class defaultSpec
```

➔ **Null Object Pattern**

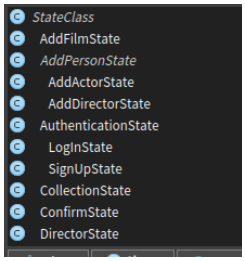
- EmptyFilm

```
    screen := ListModel new.
    screen items:  (self class currentFilm singleData).
```

```
singleData
    ^ OrderedCollection new add: 'No film has been found.';
        yourself.
```

➔ **State pattern**

- StateClass

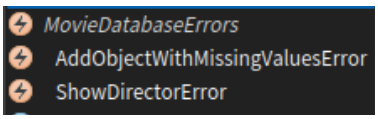



```
initializeWidgets
    state := self class state withContext: self.
```

```
dynamicChange: aStateClass
    self state: aStateClass.
    self needRebuild: false.
    self buildWithSpecLayout: self class defaultSpec
```

➔ **Custom exceptions**

- FilmDatabaseErrors
- throwing and catching : method initializePresenter in classes ViewFilmState, AddPersonState, AddFilmState, catching such exception will mostly result in ErrorState with useful information and button Return (to previous state)

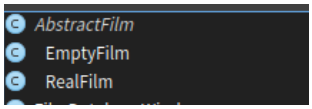



```
buttonAdd
    action: [ [ self addPerson.
        myWindow dynamicChange: self class ]
            on: AddObjectWithMissingValuesError
            do: [ :e | self goToErrorState: e ] ]
```

```
addPerson
    newPerson firstName isNil | newPerson surname isNil | newPerson born isNil
        ifTrue: [ AddObjectWithMissingValuesError signal ]
        ifFalse: [ self addNew ]
```

➔ **Singleton**

- EmptyFilm




```
new
    self error: 'cannot create instance of singleton'
```

```
instance
    ^  instance ifNil: [ instance := self basicNew initialize ]
```