

Design

PSE of

PSE Group

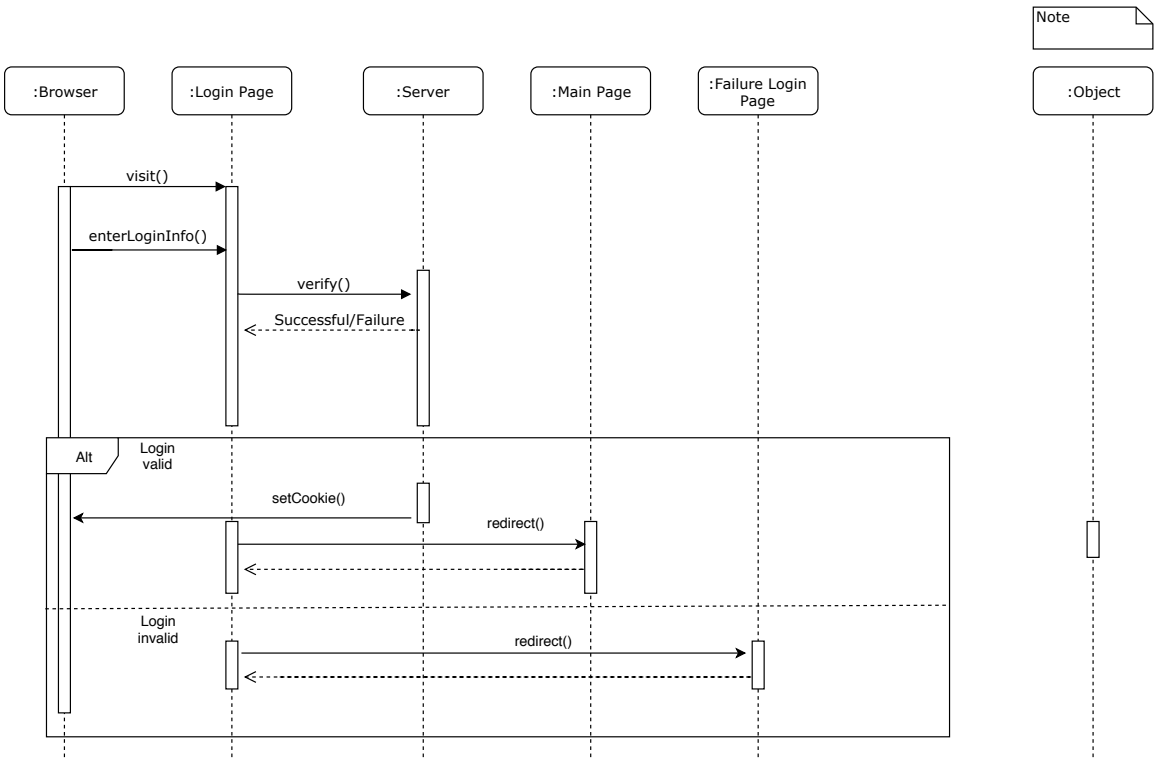
Fraunhofer Institute for Systems and Innovation Research ISI
Advisor: M.Sc. Ankush Meshram

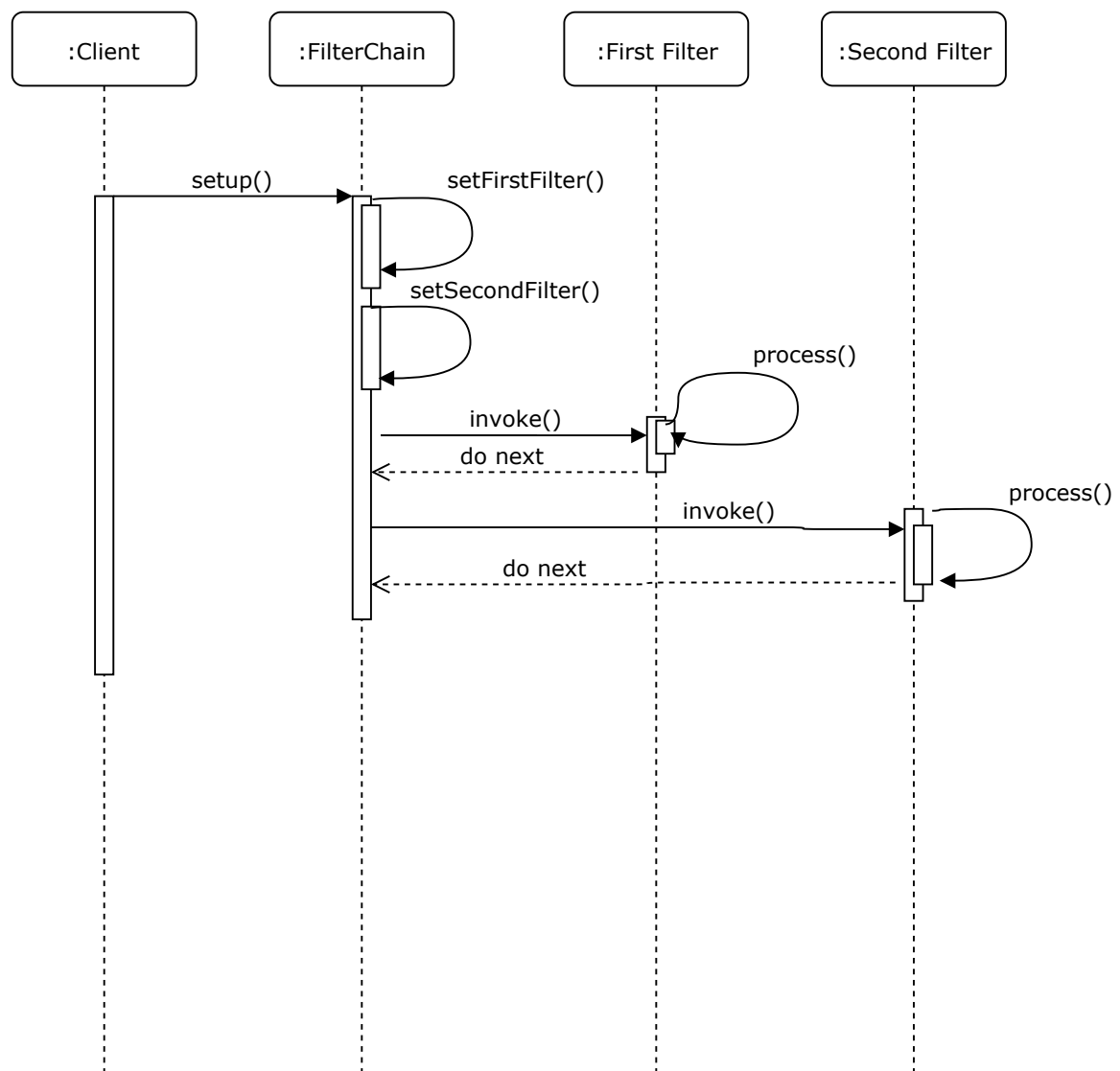
Contents

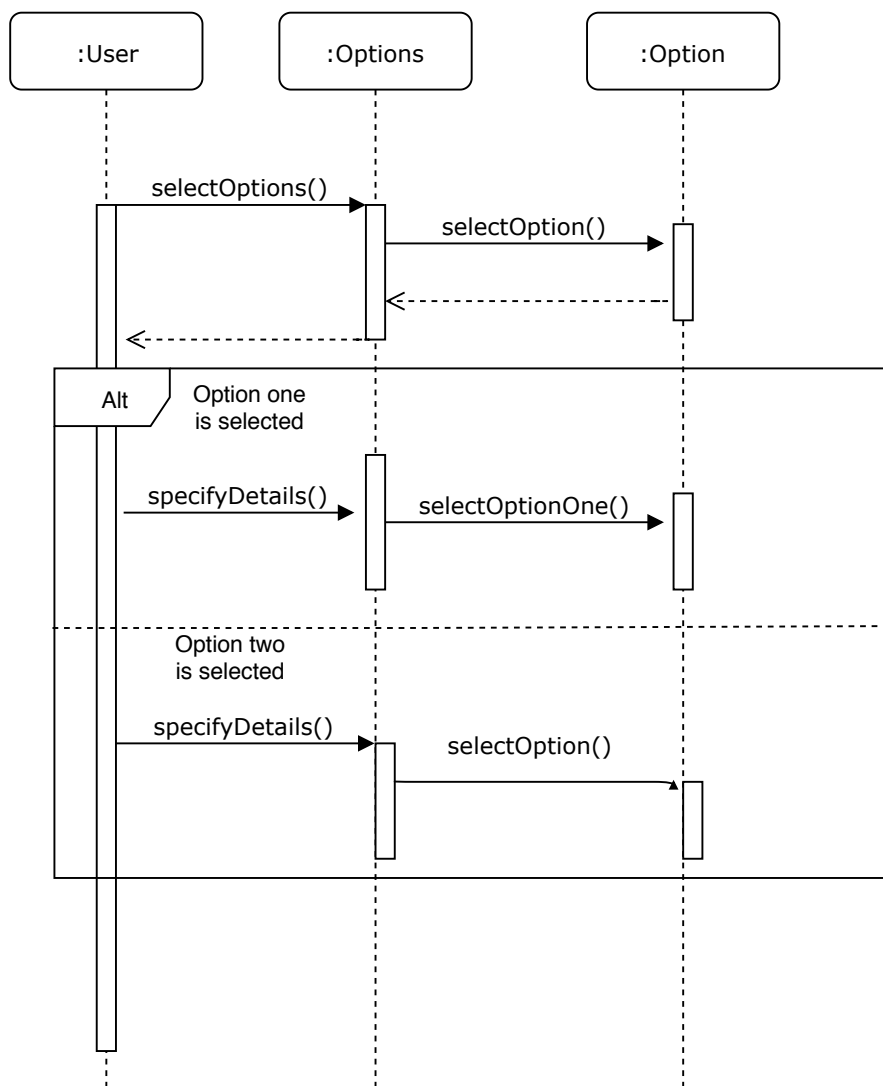
1 Design

1.1 Front-End

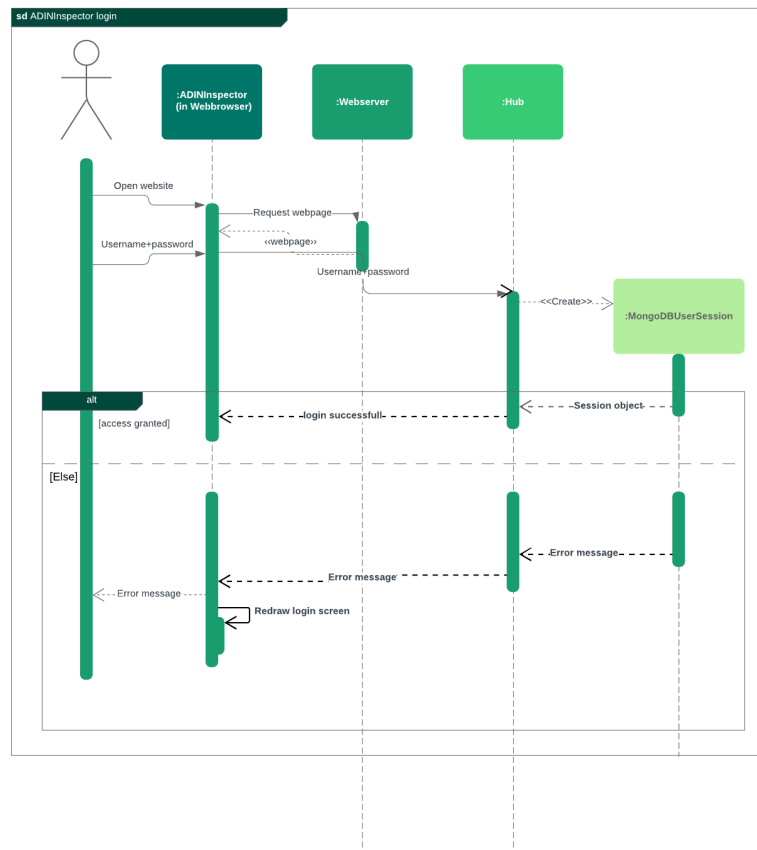
1.1.1 Sequence Diagram







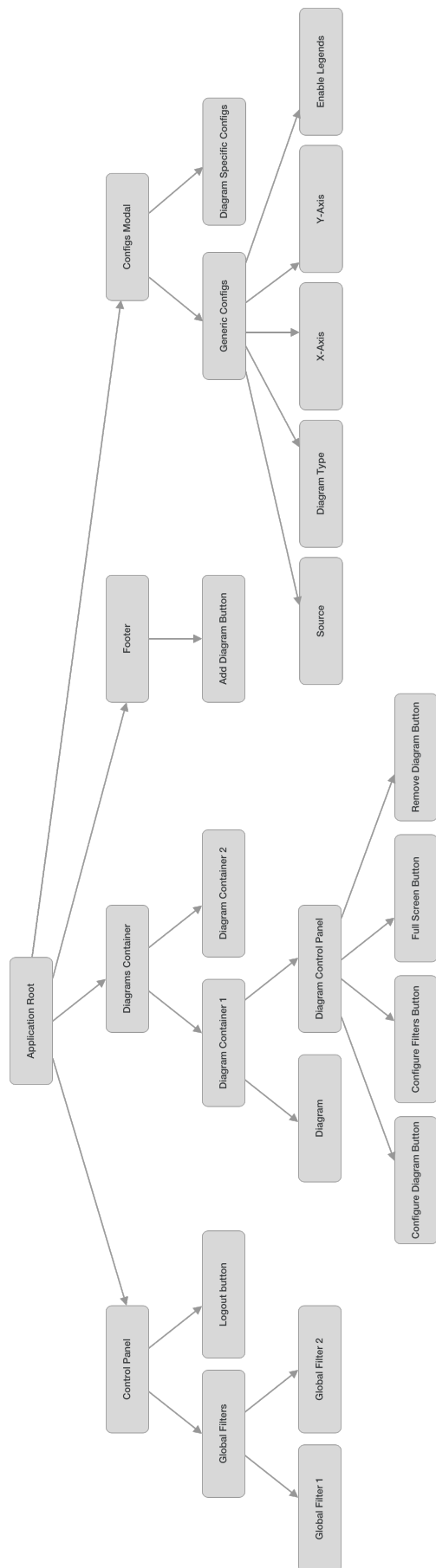
ADIN Inspector Client-Server-Communication: login



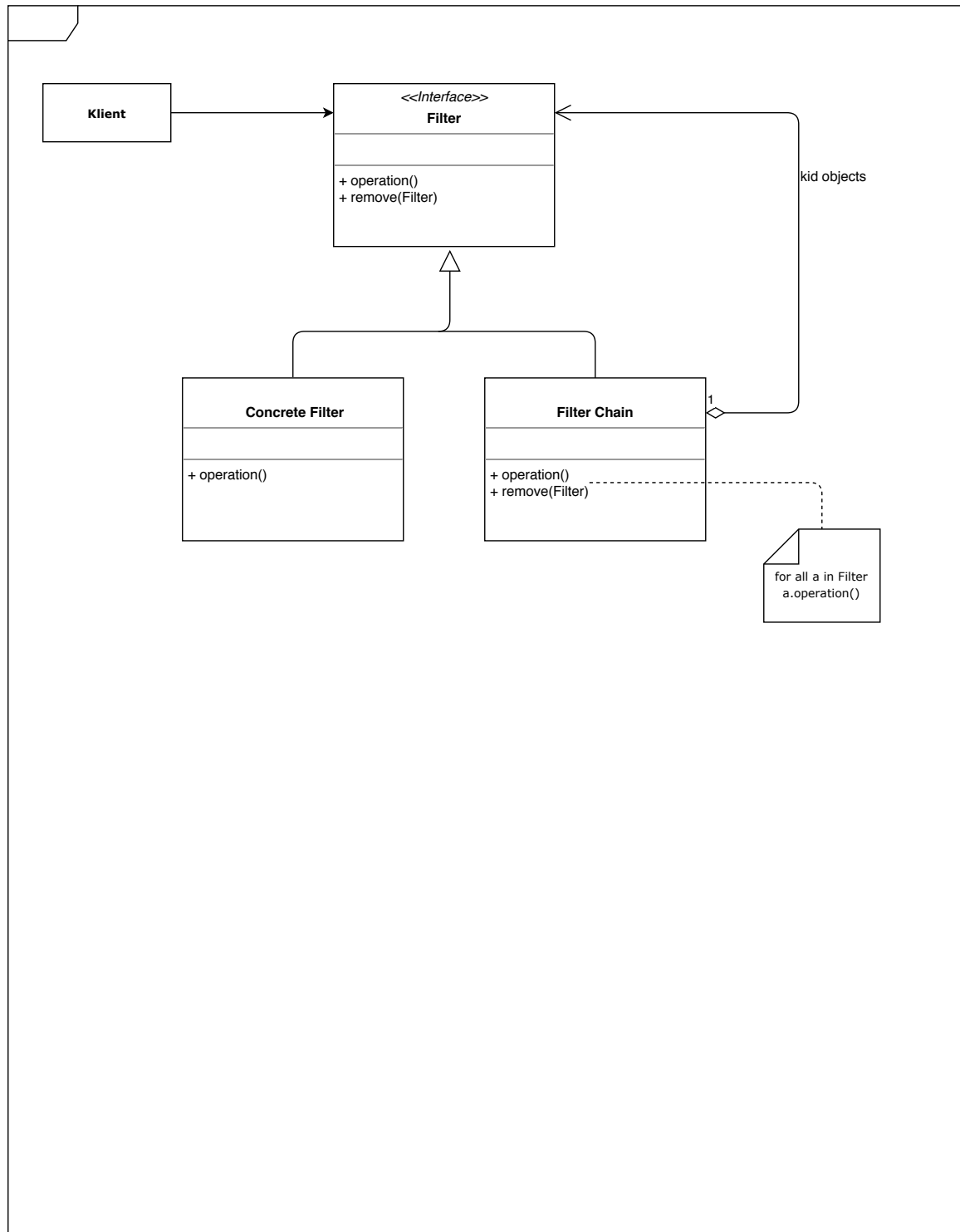
Inspector Client-Server-Communication-login.png

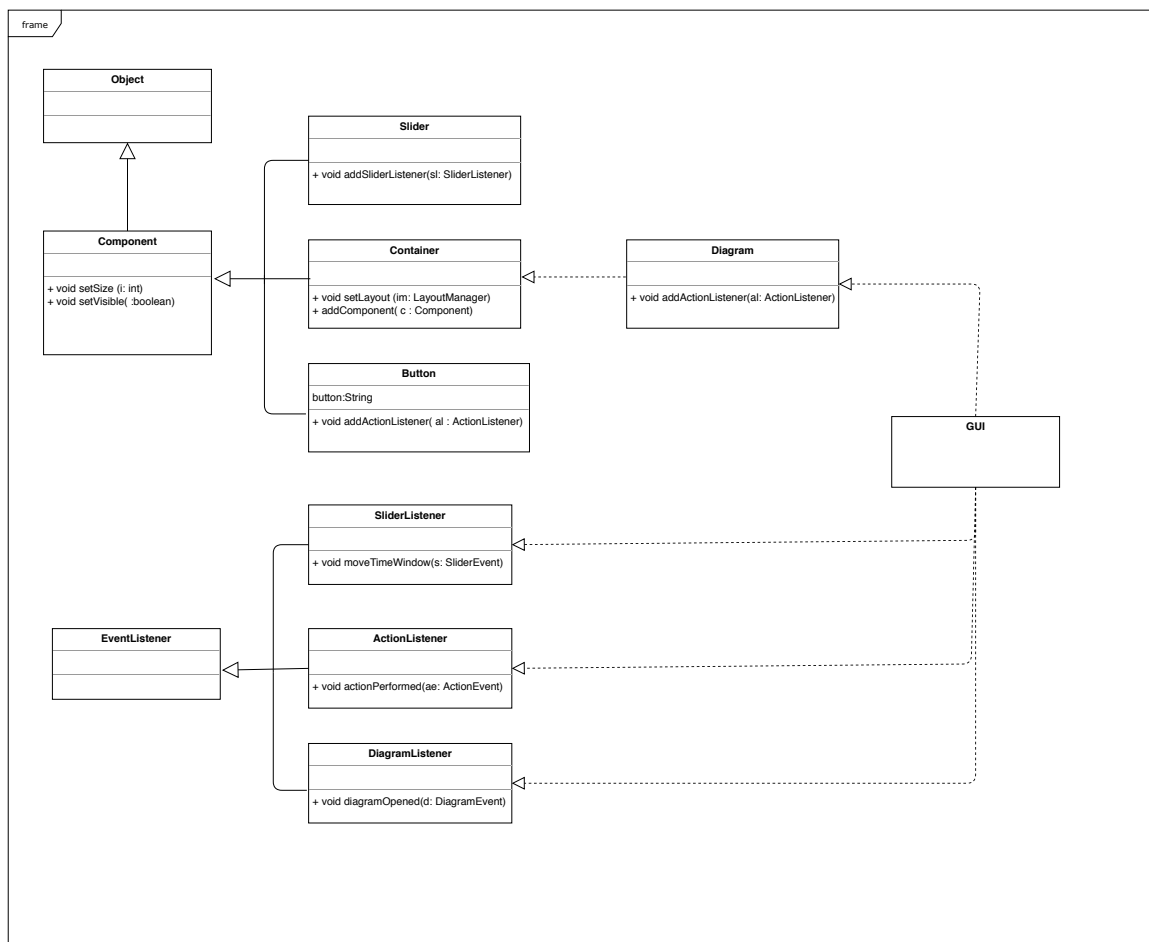
1.1.2 Activity Diagram

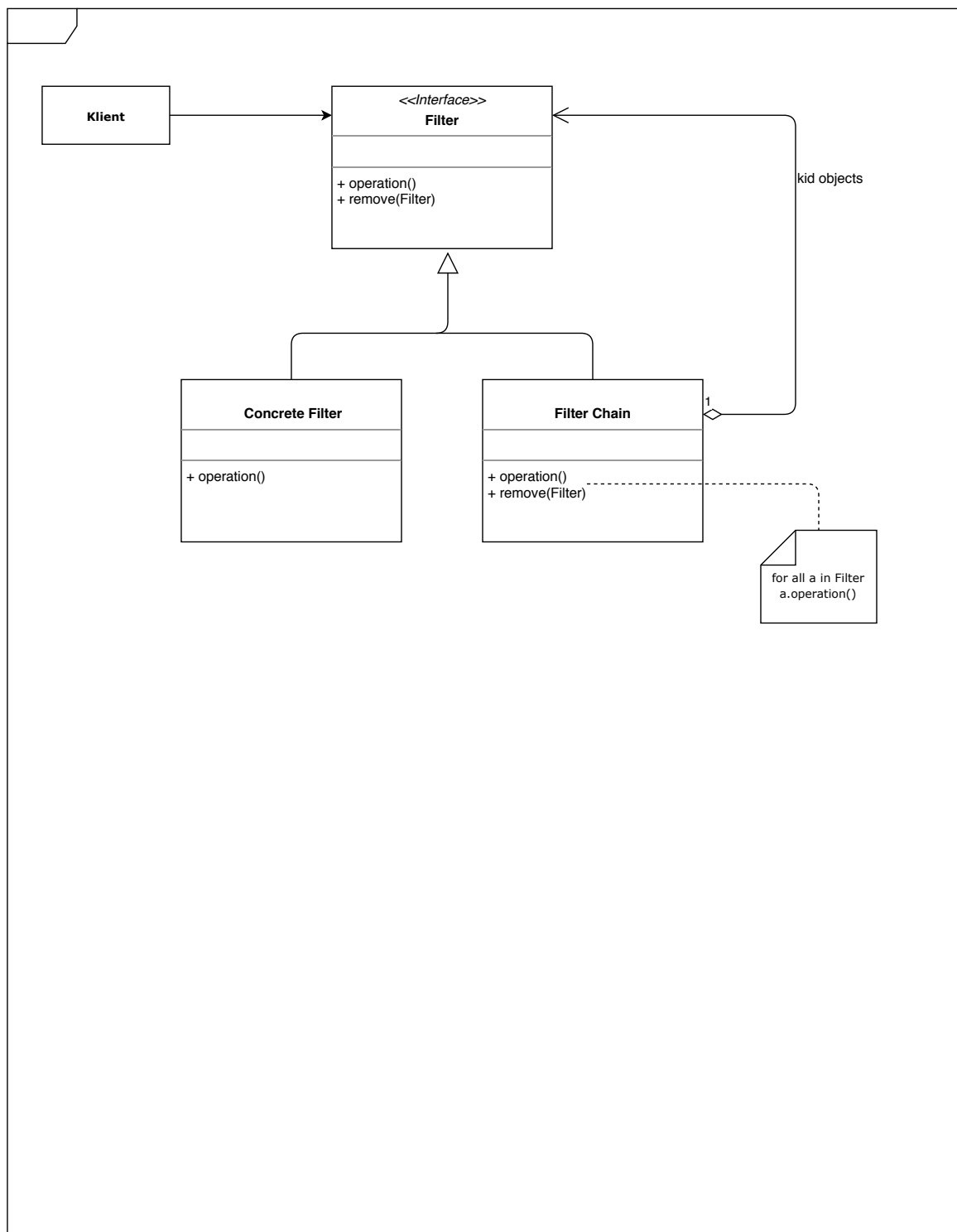
1.1.3 UI Structure Diagram



1.1.4 Class Diagram







1.2 Client-server-protocol

Messages between client and server are exchanged as strings.

Requests from client to server:

- `getAvailableCollections`

- `getCollectionSize(collection)`
- `getCollection(String coll)`
- `getCollectionInRange(key, start, end)`
- `getCollectionInRangeSize(key, start, end)`

Messages from server to client:

- list of collections: `String[]`
- collection size: `long`
- data set: `String[]` // array of json records

1.3 Back-End

This subsection deals with the back-end of the ADIN INSPECTOR. How the system deals with client http calls, and how kafka interacts with the system. An overview of the system can be seen in ??

1.3.1 Class Diagram

Next we'll look at each class and method in detail

- **Config properties file**
The config file is stored alongside the built application .jar file and contains the path to the Kafka installation folder, the user name and password of a mongoDB account with the highest level of access and the name of the database.
- **Initializer**
Methods:
 - `main`
parameters: String of arguments from the console
returns: void
App entry point.
We load the config.properties file and use the path provided to start the zookeeper, kafka and mongodb services
- **MongoConsumer**
The Mongo Consumer, as the name implies, consumes all messages from all topics in the Kafka messaging system. Once a message is found it is passed along to the Mongo Client for further processing.
Variables
 - `clientMediator`
Type : `MongoClientMediator`
An instance of the Mongo Client Mediator, created with the credentials from the config file.

Methods

- `MongoConsumer` constructor
parameters: user name and password of a mongoDB account with the highest level of access.
returns: void
Initializes the `MongoClient` variable and calls `listenForRecords()`;
 - `getAllTopics`
parameters: none
returns: an array of strings containing all the available kafka records
Asks the kafka server service which topics exists.
 - `listenForRecords`
parameters: none
returns: void
This Method first calls `getAllTopics` and uses the array of topics to poll the kafka server for new messages.
If new messages are found then the messages are passed to the Mongo Mediator for adding them to the Database.
If no new messages are found for a topic notify the Mongo Mediator that the collection tied to the topic is ready for pre-processing.
- `MongoClientMediator` This object serves as a nexus between the users who want to get data out of the database and the consumer, and `dataProcessor` who want to add data into the database.

Variables

- `client`
type: `MongoClient`
An instance of the Mongo Client from the official java API.

Methods

- `MongoClientMediator` constructor
parameters: Username and password
returns: void
Initializes the client variable, throws an error if the user is not found.
- `addRecordToCollection`
parameters: String representation of a record in json format
String name of the collection it should be added to.
returns: void
Converts the json string into a java object, then to a bson document and uses the mongoAPI to insert it into the database.
- `addRecordsToCollection`
parameters: String Array of records to be added to a collection
String name of the collection it should be added to.
returns: void
for each one of the members of the array call `addRecordToCollection`

- ProcessCollection QUESTIONS FOR ANKUSH
parameters: String, name of a collection
returns: void
 - getCollection
parameters: String, name of a collection
returns: String array containing all entries of the collection
 - getStartRecord
parameters: String, name of a collection
returns: the first entry of the collection as a String.
 - getEndRecord
parameters: String, name of a collection
returns: the last entry of the collection as a String.
 - getCollectionSize
parameters: String, name of a collection
returns: the number of entries in the collectoin as int
 - getCollectionInRange
parameters: String, key of the parameter used for filtering
String start and end ranges for the filtering
returns: String array containing all entries of the collection within that range
this Method is very general to allow for flexibility.For example by letting the key be, SourceIPaddresses, or a timeStamp.
 - getCollectionInRange
parameters: String, key of the parameter used for filtering
String start and end ranges for the filtering
returns: number of elements matching the range as int
this Method is very general to allow for flexibility.For example by letting the key be, SourceIPaddresses, or a timeStamp.
- Record
Every message that comes from kafka and needs to be added to the database has it's own Record class that inherit from this one.
Every single class that inherits needs to be able to, using reflection, convert itself into a Bson Document where every variable is a key Value pair of the name of the variable and it's associated value.
Variables
 - id
type: StringMethods
 - getAsDocument()
parameters: none
returns: A Document, containing every variable of any class inheriting from this one.

This function checks for every variable, gets it's name and value as a string and adds it to the document that it eventually returns.

- PacketRecord

Inheriting from Record, this class contains the variables that match the json string obtained from kafka.

Variables

- id
type: String
this id is used for determining the ordering when saving to mongoDB, it's the offset of the message in the kafka messaging queue. inherited from Record
- client
type: String
- L2Protocol
type: String
- SourceMACAddress
type: String
- L4Protocol
type: String
- SourceIPAddress
type: String
- PacketSummary
type: String
- DestinationIPAddress
type: String
- Timestamp
type: String
- DestinationPort
type: String
- SourcePort
type: String
- DestinationMACAddress
type: String

Methods

- getters / setters
parameters: none
returns: variable type
Each variable has it's getters and setter methods.

- AlarmRecord

Variables

- method

An instance of the Mongo Client from the java API

Methods

- getters / setters

parameters: none

returns: variable type

Each variable has it's getters and setter methods.

- AggregatedRecord

Variables

- method

An instance of the Mongo Client from the java API

Methods

- getters / setters

parameters: none

returns: variable type

Each variable has it's getters and setter methods.

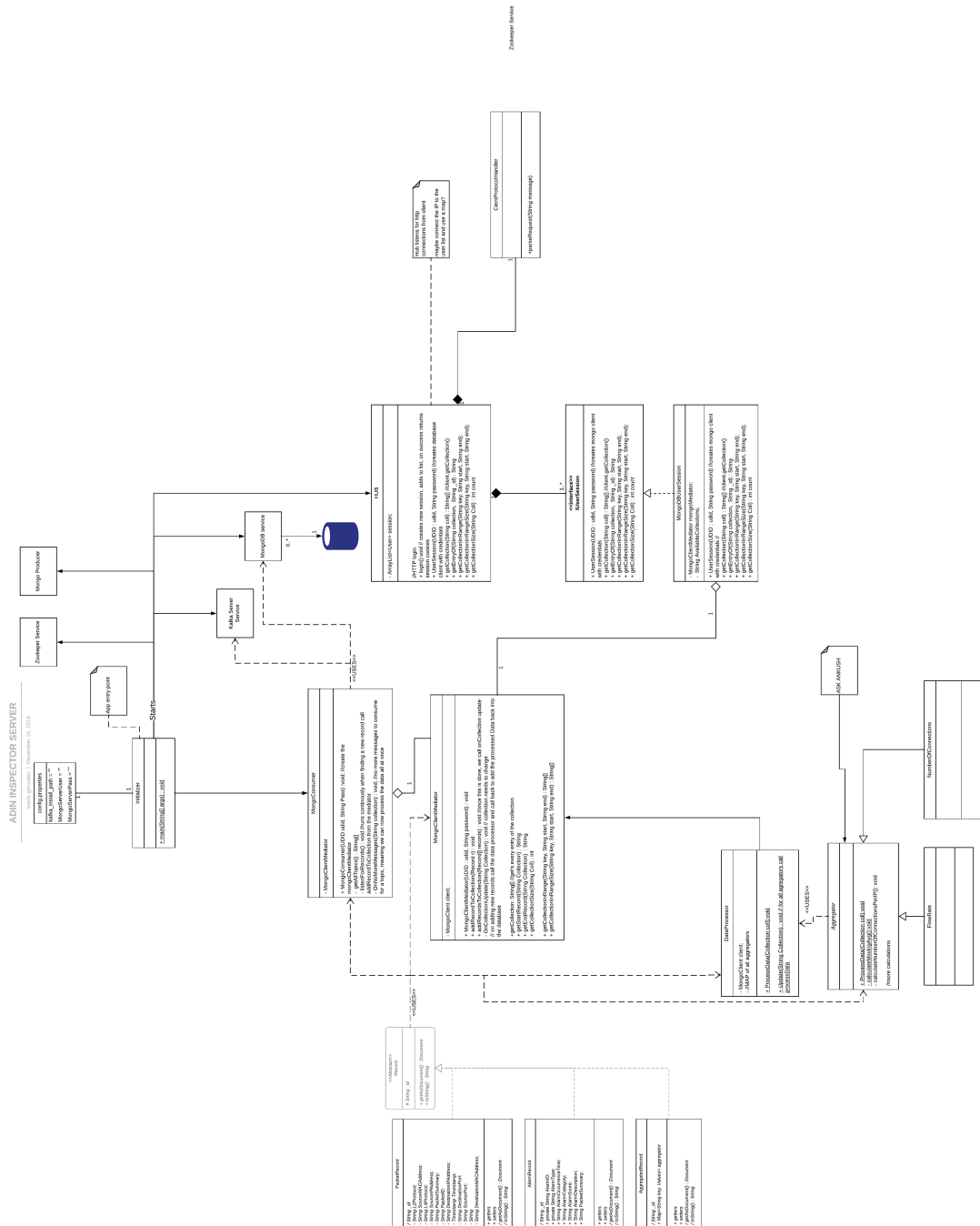
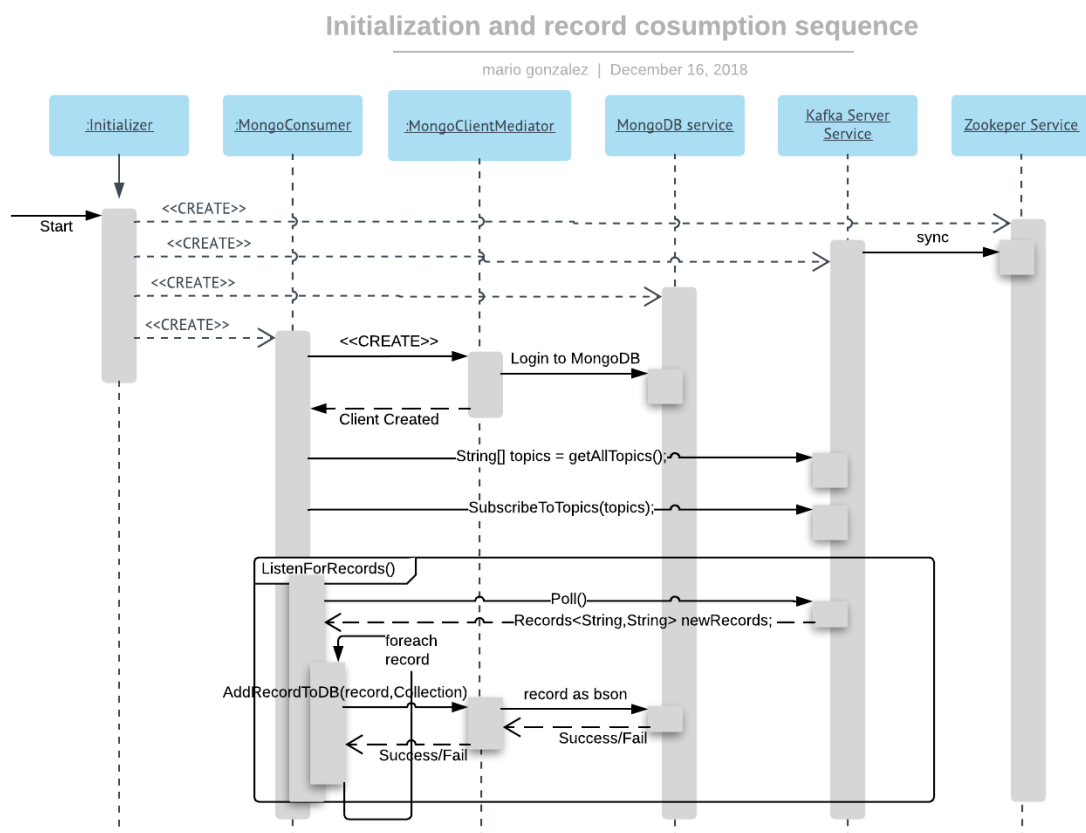


Figure 1: This is the class diagram for the whole back-end system

1.3.2 Sequence Diagram



1.3.3 Activity Diagram