Contents

[User requirements 2](#_Toc74268002)

[The use case diagram 3](#_Toc74268003)

[The class diagram – analytical 3](#_Toc74268004)

[The class diagram – design 4](#_Toc74268005)

[The scenario(get all the buildings in the selected settlement) 4](#_Toc74268006)

[The activity diagram 5](#_Toc74268007)

[The state diagram 6](#_Toc74268008)

[The interaction diagram 7](#_Toc74268009)

[The GUI design 8](#_Toc74268010)

[The discussion of design decisions and the effect of dynamic analysis 11](#_Toc74268011)

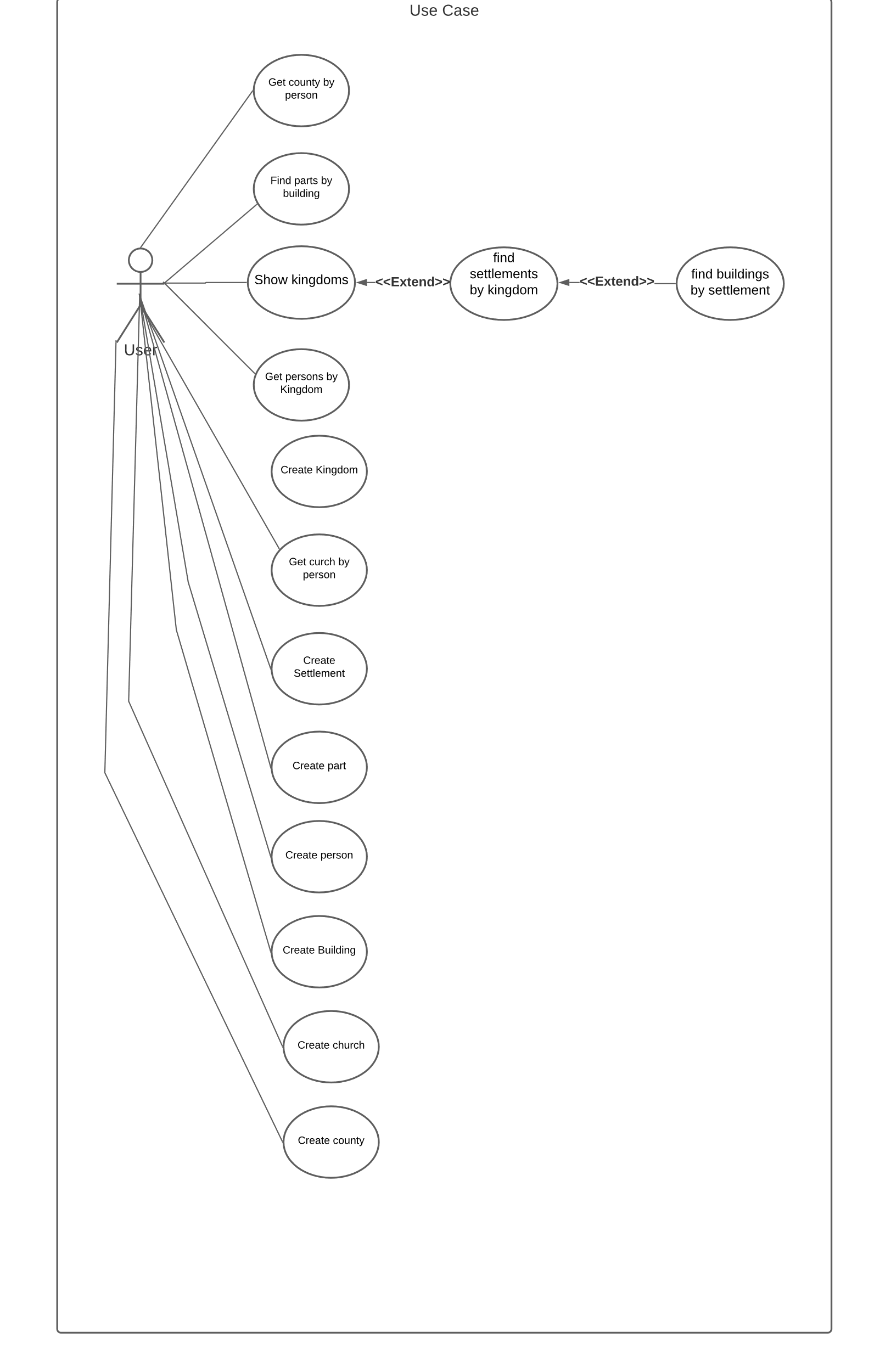
# User requirements

The gaming company needs a program to store data about the Kingdoms for further use in future projects.

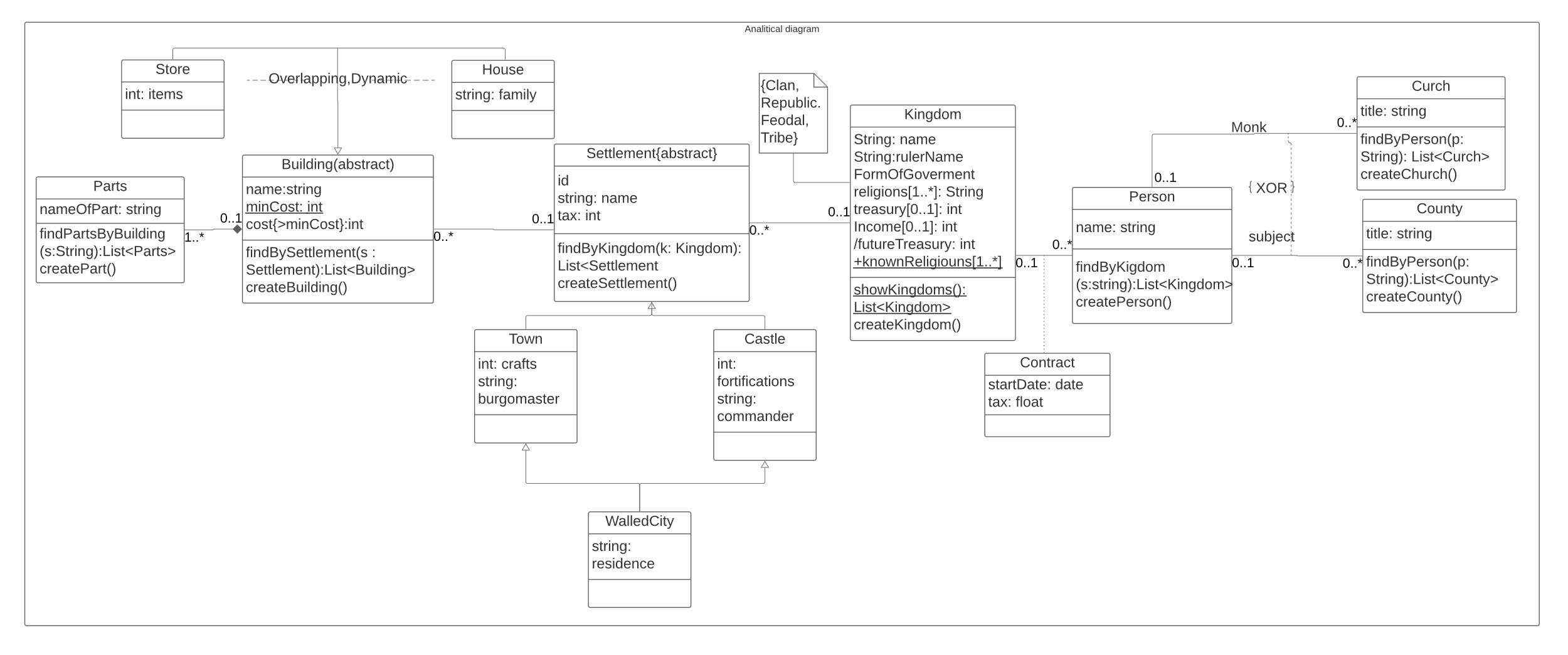
Each kingdom has a name, and name of a ruler also, each kingdom has one of the standard forms of government (Clan, Republic. Feodal, Tribe). There can be many religions in the kingdom, but there must be at least one that, for example, is professed by the ruler. For all kingdoms there is also a list of known religions in this list there must be at least one such religion. Each kingdom has its own treasury and the income that the kingdom receives each year and from these indicators is calculated the future treasury of the kingdom. Every kingdom has a persons and every person has a name. These persons enter into a contract with the kingdom. The contract includes the start date of the contract and the tax that the person must pay. Everyone in the kingdom can be either a county subject or a monk in the church. There is no option where a person can be a subject and a monk at the same time. Each church and county have name. There are many settlements in the kingdom. Each settlement has a name, a tax and its own unique identifier. Settlements are different: castles and cities. The castle has a number of fortifications and a commander. The city has a number of craft and burgomasters. There is also a settlements that combine both the castle and the city they are called walled city and such a city has a residence. In each settlement there are buildings, they have a name and the cost of maintenance which cannot be less than the minimum cost of maintenance for all buildings. There are different types of buildings: shops and ordinary houses. For the store you need to keep the number of goods, and for the house the name of the family that lives there. The building can be at the same time both shop and the house and can change the specialization for example from shop to the usual house or from house to the shop. Each building is made up of parts. Each part has its own name.

1. The user must be able to get a list of kingdoms.
2. The user must be able to get all the information about the selected kingdom and then user can get settlements with all information about infrastructure of the settlement.
3. The user must be able to get all persons in the selected kingdom.
4. User must be able to create a Kingdom.
5. User must be able to create a Person.
6. User must be able to create a Church.
7. User must be able to create a Building.
8. User must be able to create a Part.
9. User must be able to create a Settlement.
10. User must be able to create a County.
11. User must be able to find a church by person’s name.
12. User must be able to find a county by person’s name.
13. User must be able to parts of a selected building.

# The use case diagram



# The class diagram – analytical



# The class diagram – design

# 

# The scenario(get all the buildings in the selected settlement)

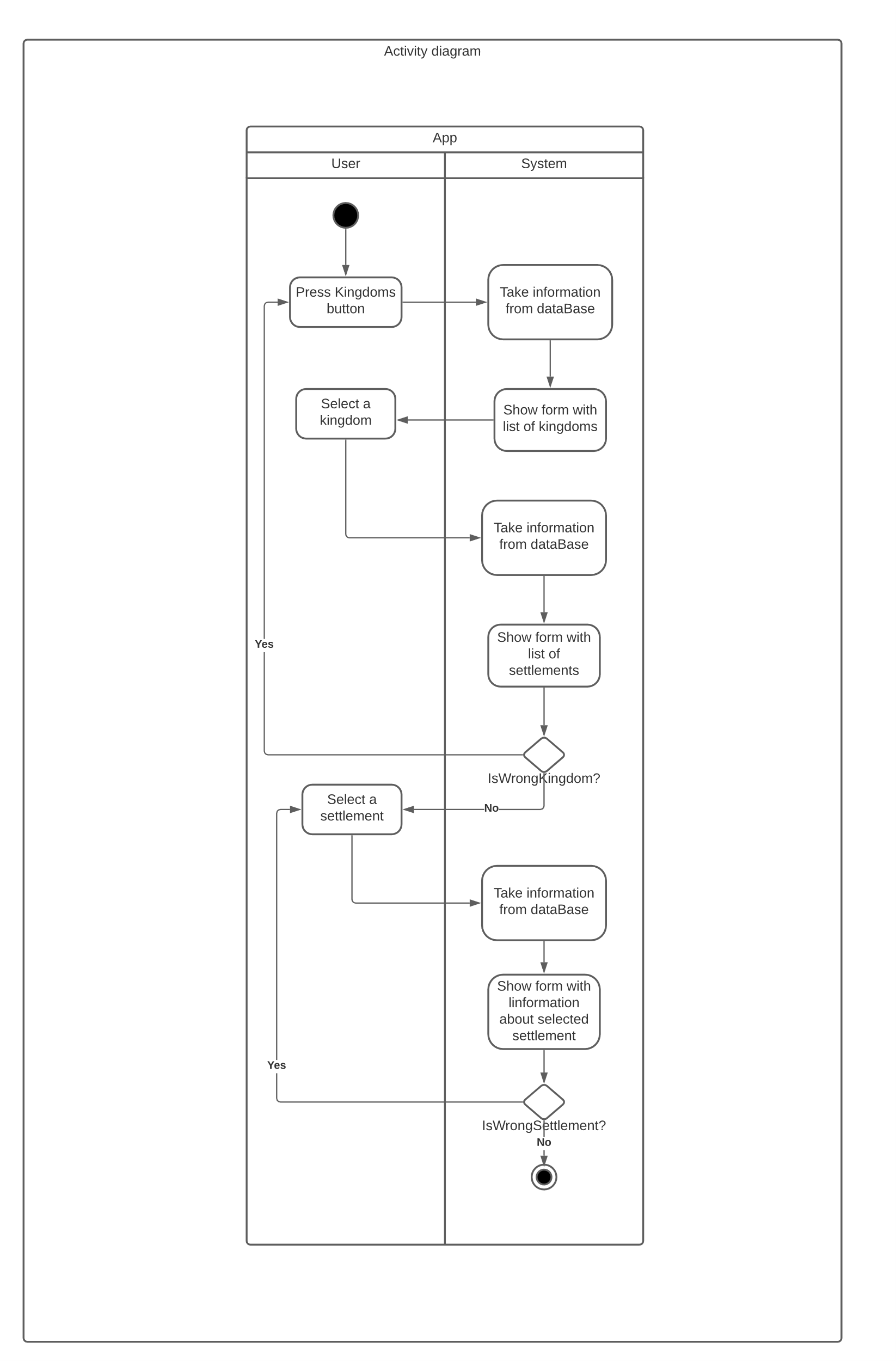
Main scenario:

1. User press the button to get list of kingdoms.
2. System fetch data from a database.
3. System shows a form with a list of kingdoms.
4. User browses though list of kingdoms and selects one of them.
5. System fetch data from a database.
6. System show form with a list of settlements and information about selected kingdom.
7. User get list of settlements and information about selected kingdom.
8. User browses though list of settlements and selects one of them.
9. System fetch data from a database.
10. System show form with all information about infrastructure in the selected settlement.
11. User get all the buildings in the selected settlement with information. about each building at the bottom of the screen.

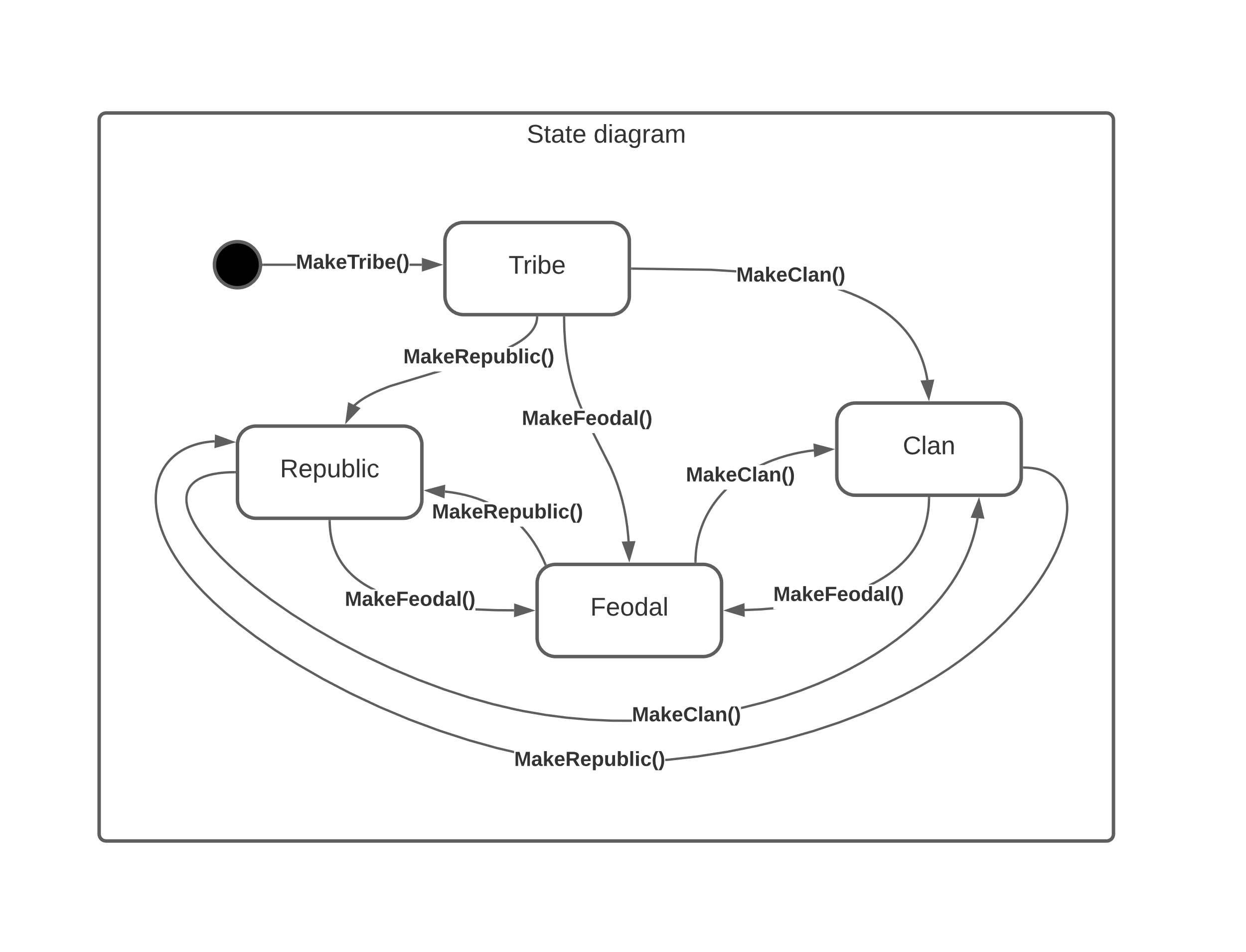
Alternative scenario(wrong choice of kingdom):

1. User press the button to get list of kingdoms.
2. System fetch data from a database.
3. System shows a form with a list of kingdoms.
4. User browses though list of kingdoms and selects one of them.
5. System fetch data from a database.
6. System show form with a list of settlements and information about selected kingdom.
7. User press the “kingdom” button to return to the list of kingdoms
8. System fetch data from a database.
9. System show form with a list of kingdoms.
10. User browses though list of kingdoms and selects one of them.
11. System fetch data from a database.
12. System show form with a list of settlements and information about selected kingdom.
13. User get list of settlements and information about selected kingdom.
14. User browses though list of settlements and selects one of them.
15. System fetch data from a database.
16. System show form with all information about infrastructure in the selected settlement.
17. User get all the buildings in the selected settlement with information. about each building at the bottom of the screen.

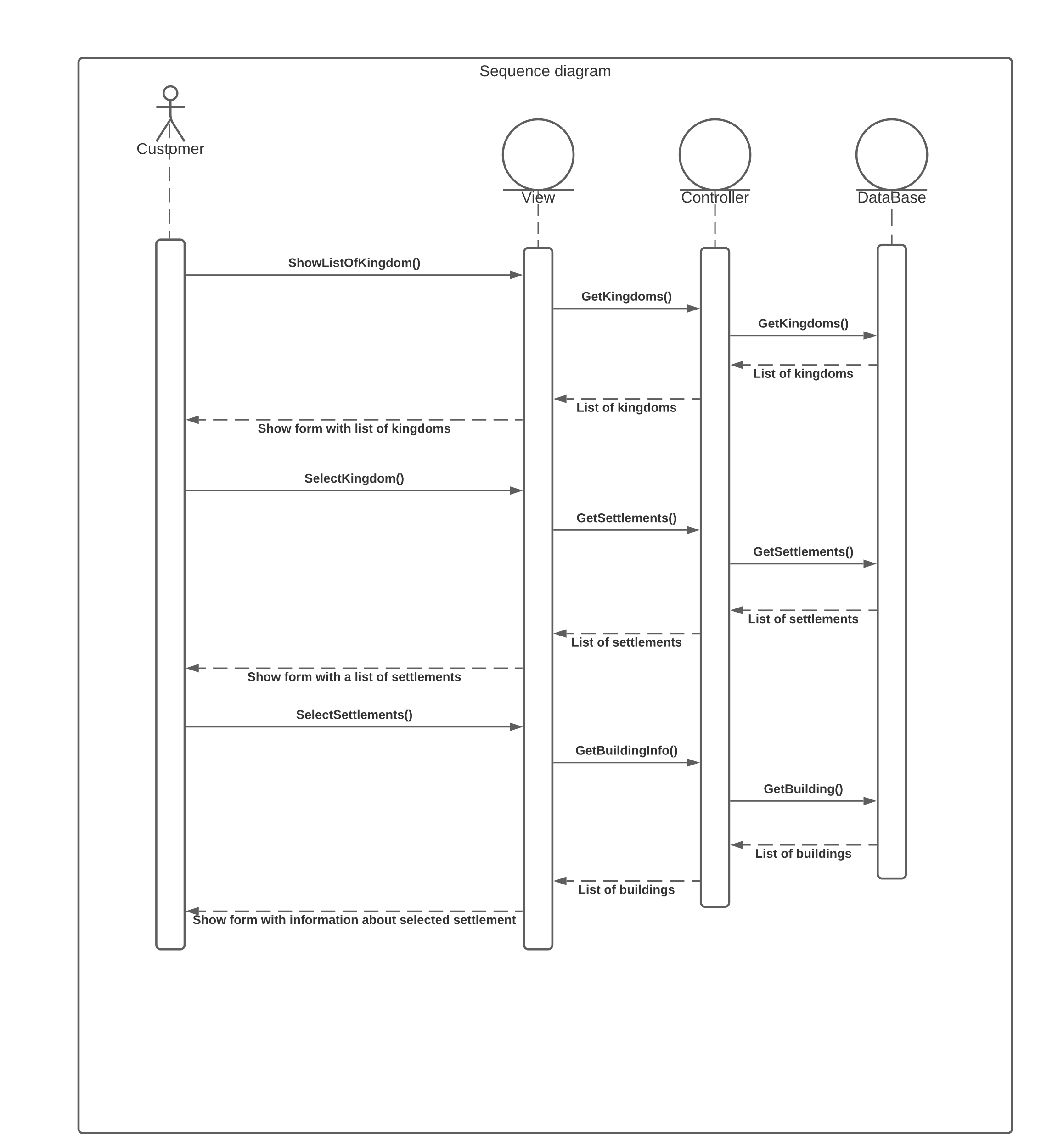
# The activity diagram



# The state diagram



# The interaction diagram

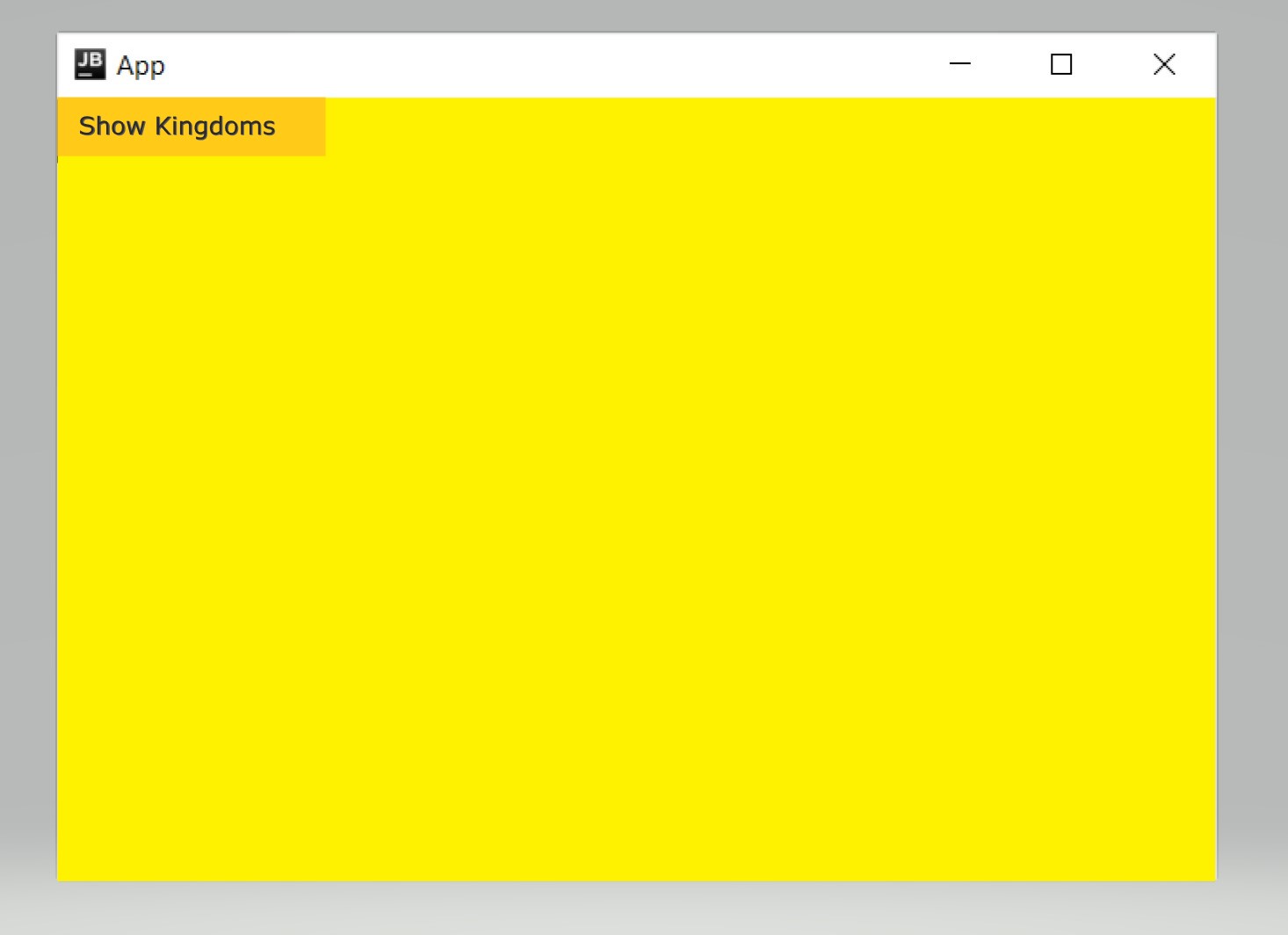


# The GUI design

Here I will describe an approximate interface design

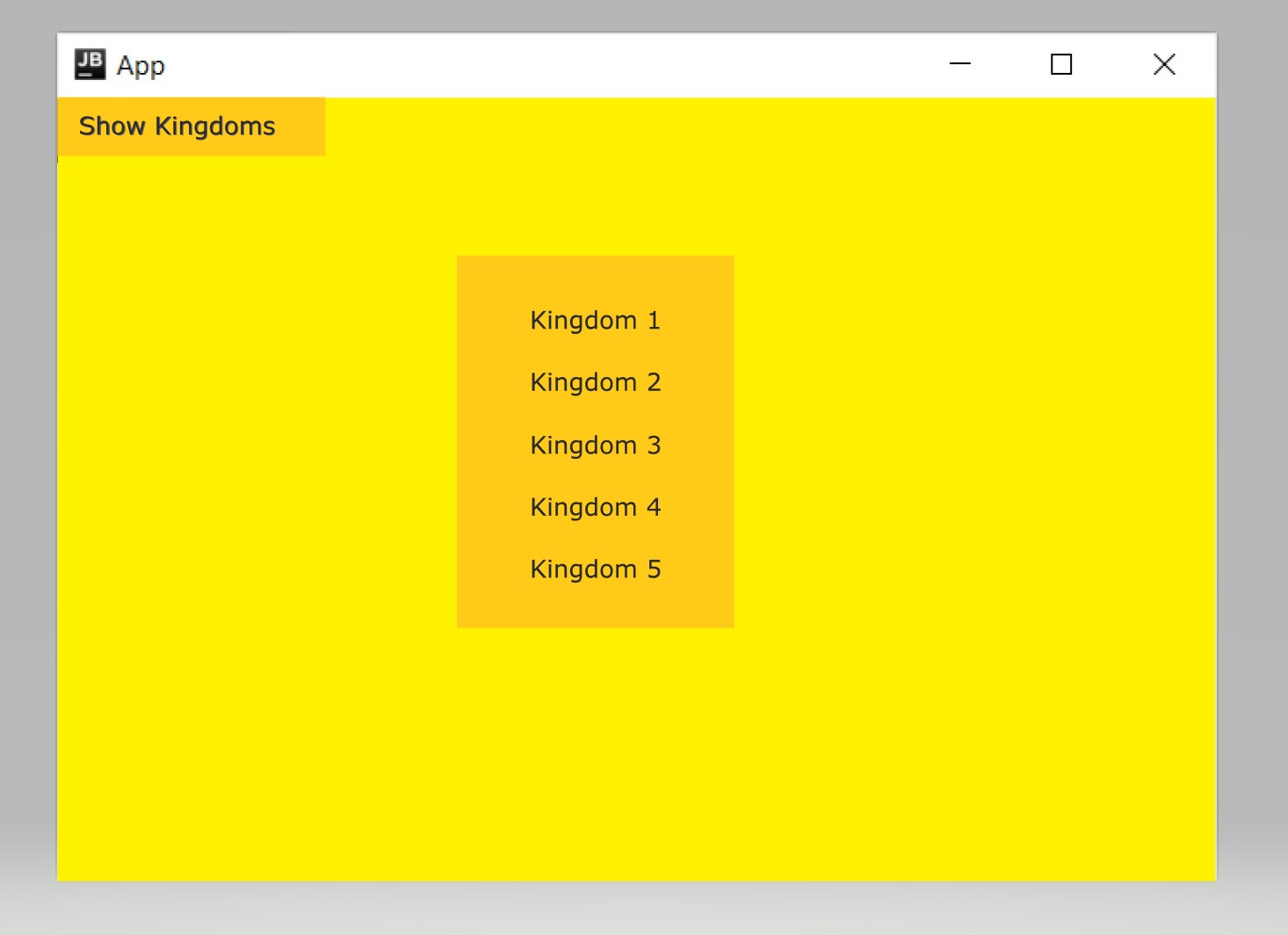
1.Start form

On the home page we can get a list of kingdoms.



2. Kingdoms form

And in this form we can choose the kingdom that interests us. By choosing a kingdom we can see all the settlements in that kingdom and all information regarding that kingdom.



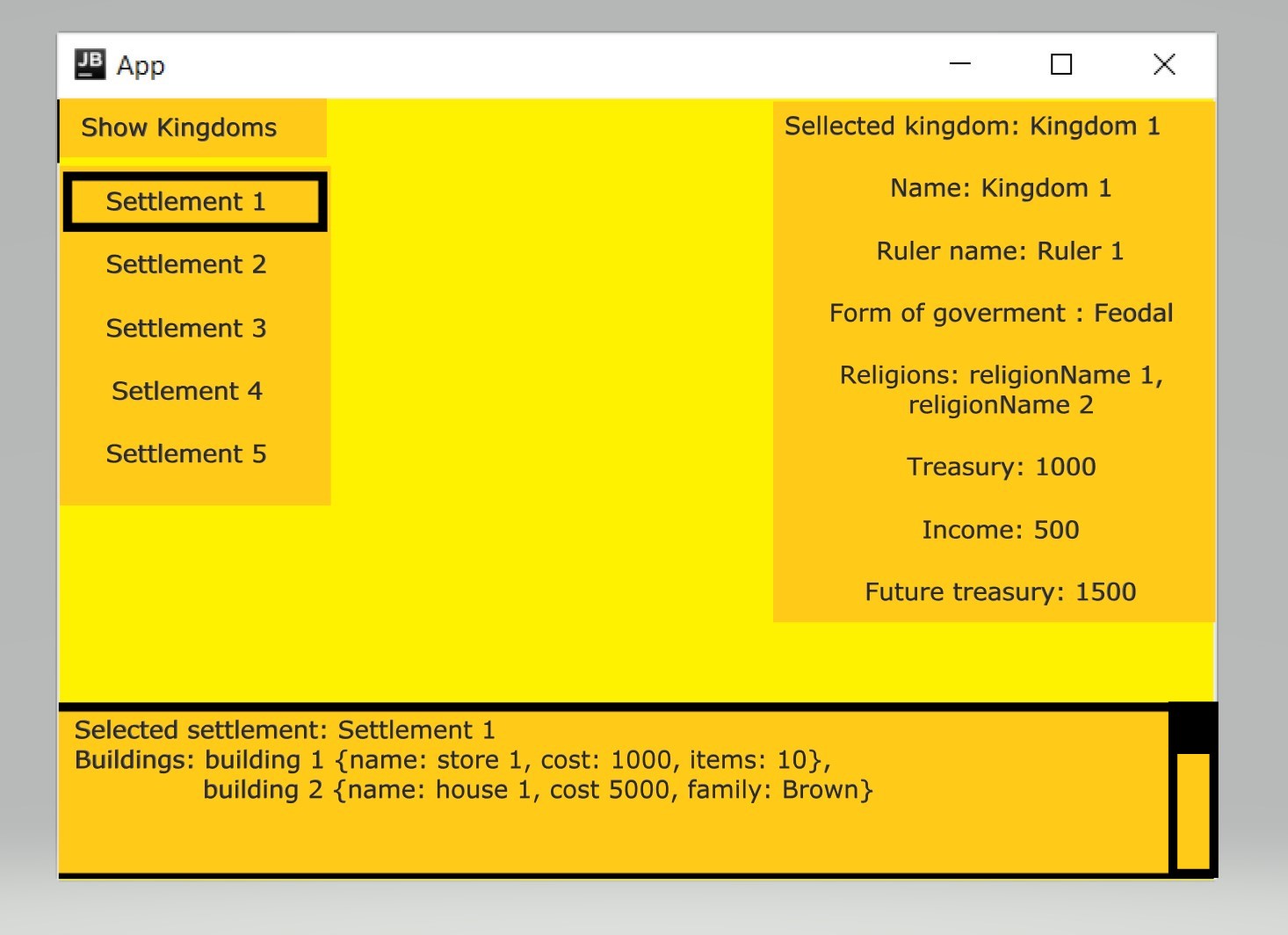
3. Settlements form

All settlements in this kingdom. Here we can choose settlement that we need.



4. Building form

After selecting a settlement below will be all the information about the infrastructure of the settlement: all the buildings and information stored in them.



# The discussion of design decisions and the effect of dynamic analysis

The application will be created using Java, Lombok, Hibernate.

Each class has a repository that uses a class extent in methods. These methods can be used to find certain information. In all classes, the creation methods were replaced by the methods used in the lombok for the same purpose(createPerson() -> +buld(),+builder()).

Kingdom class:

In the class kingdom is implemented enum class. This enum is responsible for the form of government in the kingdom(state diagram). Types of multi-value attributes were also defined. For both multi-value attributes in this class will be used Set of strings (Set<String>), because order does not interest me and I need only a collection to store an objects.

Also in Kingdom class was added methods from state diagram(makeTribe(),makeClan(),makeFeodal(),makeRepublic()) and from sequence diagram(selectKingdom(k: Kingdom)).

Settlement classes:

In Settlement class was added method from sequence diagram(selectSettlement(s:Settlement)). Settlement is an abstract class. And there are 2 classes that extend the class Settlement (Castle and Town). There is also a class that extends these two classes (WalledCity). In Java there is no possibility to make such inheritance directly. It is a multi-inheritance, and I implemented this inheritance by adding interface (ICastle) and implementation class (CastleImp). CastleImp class extends the super class and implements ICastle interface. Town class extends only a super class. Last class WalledCity extends a Town class and implements ICastle interface. Thus, the WalledCity class has the properties of both the city and the castle.

Building classes:

In this case, as we can see in the analytical diagram I need to implement overlapping and dynamic inheritance. As last time, this type of inheritance cannot be directly implemented in Java. So, I created 2 interfaces instead of 2 classes in analytical diagram (House class -> House interface, Store class -> Store interface). My Building class implements both this interfaces and has an enum(BuildingType). This enum represent kind of status of building. Building can have a property of both interfaces or only one and can change properties if it needs. To change the properties, I created two methods(becomeStore() and becomeHouse()).