Online show Booking App

Software Engineering Lab WS22/23

Ibrahim Alsabbagh (3104154)

Supervisor: Marcel Schweikert

Date of submission: 25.10.2022

A1

Requirements

- R1: In order to book a cinema ticket, customers must first register using a unique email address and a password.
- R2: Customers can then log into their customer account, also by entering their email address and password.
- R3: Customers can log out of the website when they want.
- R4: Customers should be able to display a list of future shows. This list should be sorted in ascending order of the time of the shows (next first) and each show should be clearly identifiable by an ID.
- R5: The employees of the cinema should be able to create new shows by providing the necessary attributes.
- R6: Registered customers can book tickets for a show by selecting it from the list and then specifying how many and which seats they want.This will only be possible to do no later than 15 minutes before the start of the show.
- R7: After a show has started, it should be marked as "archived" so that it is still available to the staff, but no longer appears in the list of shows.
- R8: Employees should also be able to cancel shows and their bookings and delete them from our database which also Informs the customers by sending emails to them in case of cancellation of the show.
- R9: All shows that are more than 1 year old should also be automatically removed from the system together with all associated bookings.

Assumptions

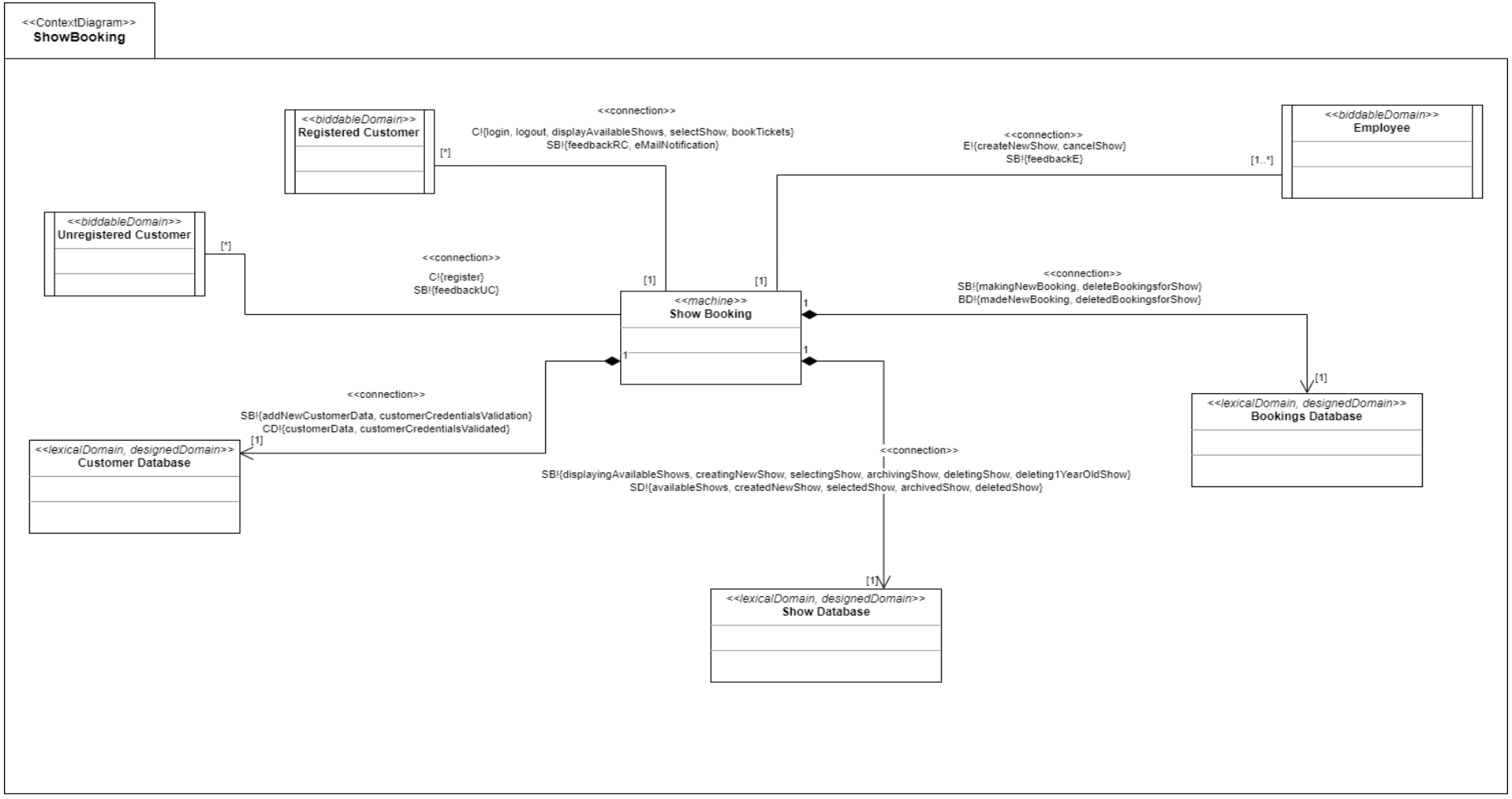
- A1: Customers will only provide email addresses to which they have access themselves.
- A2: Customers will regularly check this website for new shows.
- A3: Webapp is a suitable choice and can be used by everyone.
- A4: Every booking will be paid for.
- A5: Employees will only create shows that will take place in the future.

Facts

- F1: A show consists of the title of the film, the duration, an assigned hall and a date (date & time) on which the film will be shown.
- F2: A hall consists of a predetermined number of rows, the number of seats per row and a unique hall number.

A1

Context Diagram

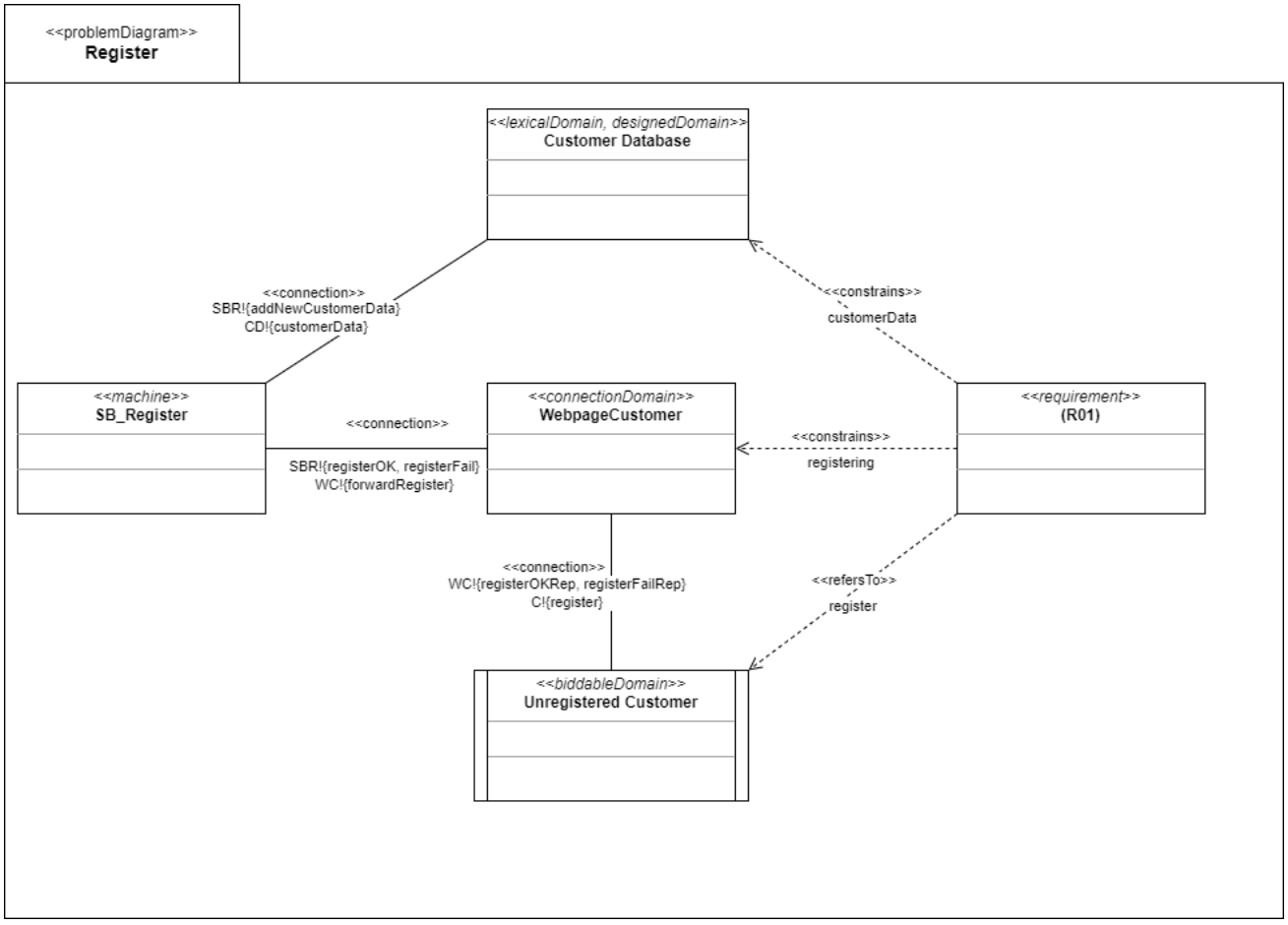


A2

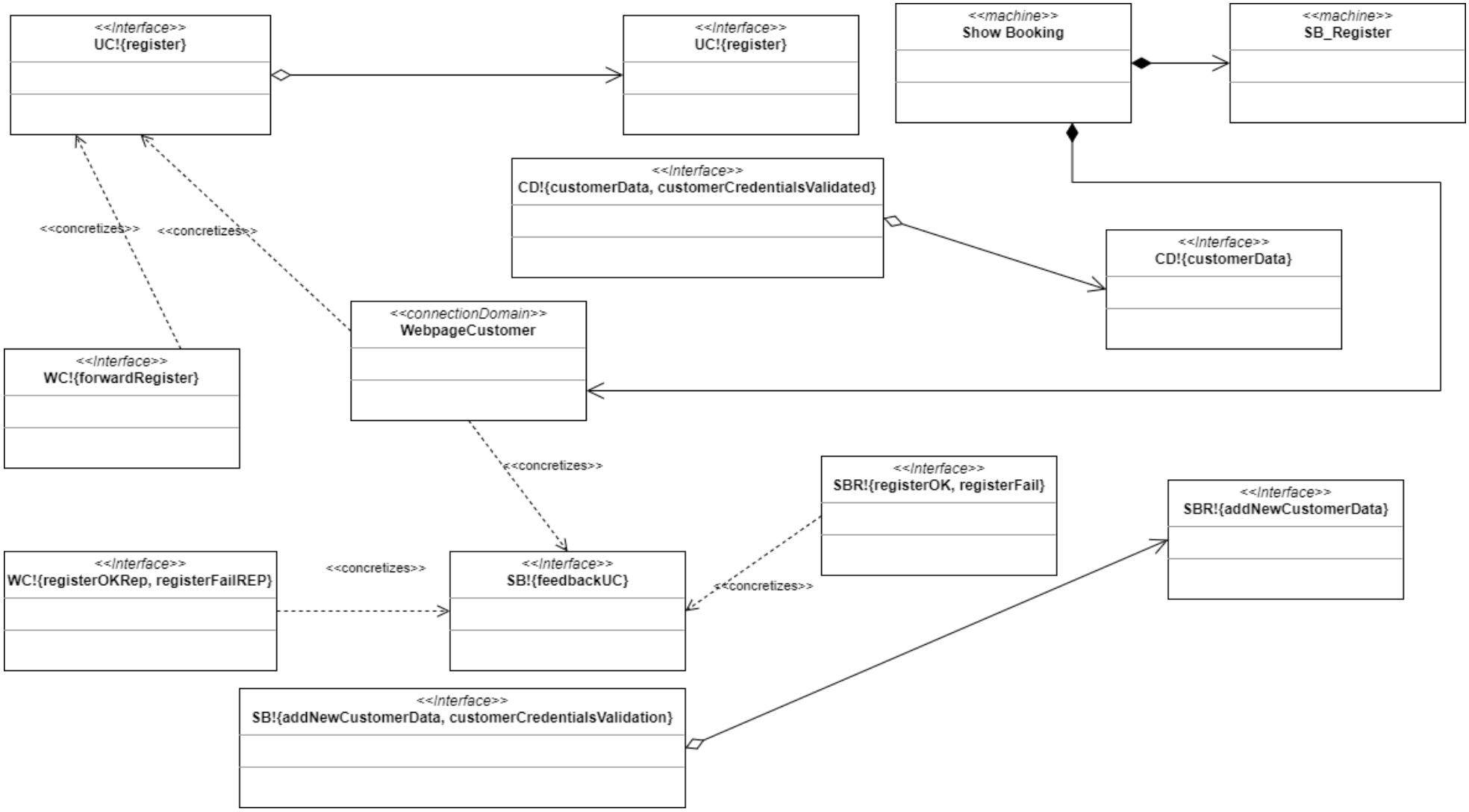
Problem Diagrams

1. In order to book a cinema ticket, customers must first register using a unique email address and a password.

For this requirement we used the problem frame **update II**

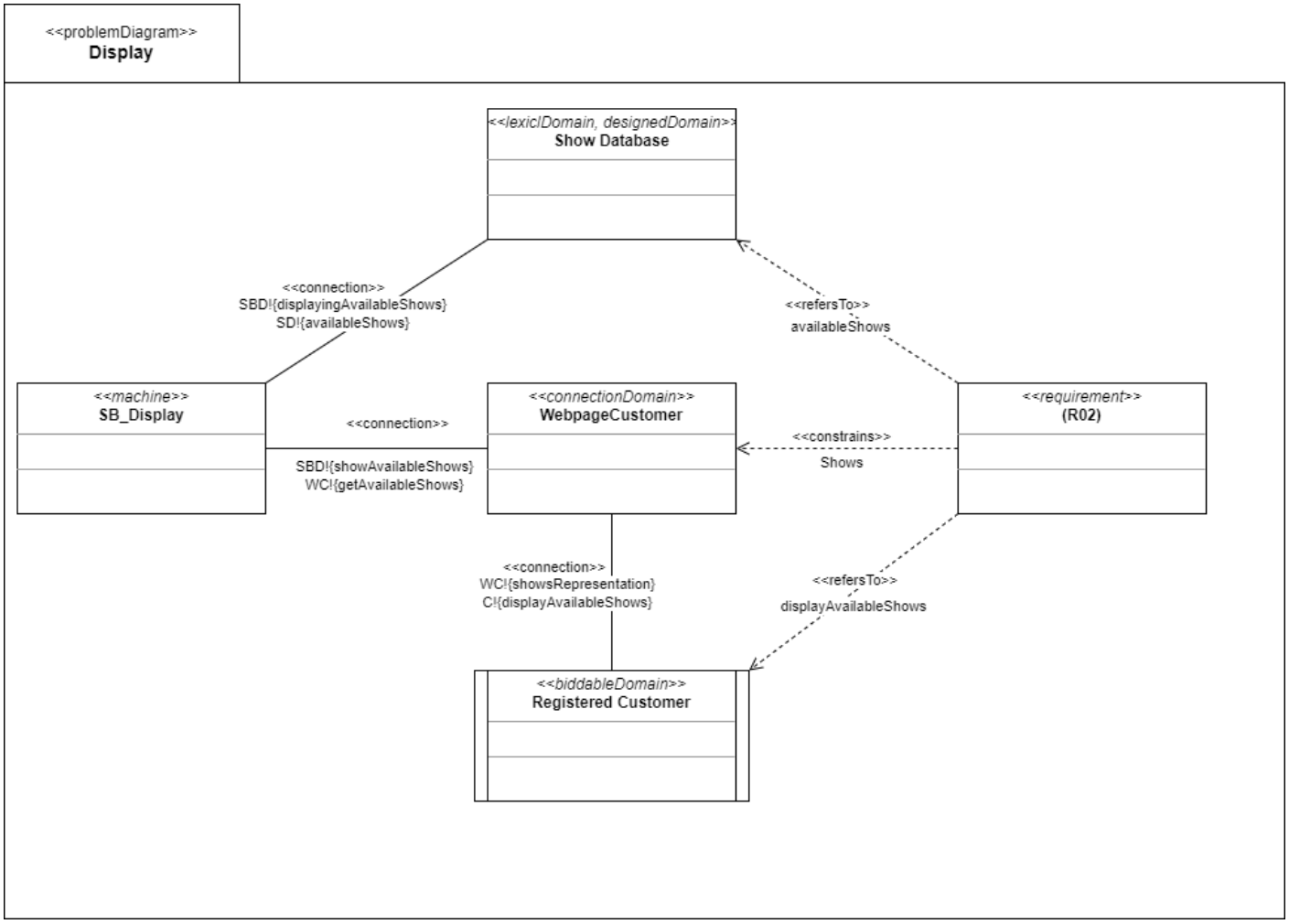


Concretize Interface(s):

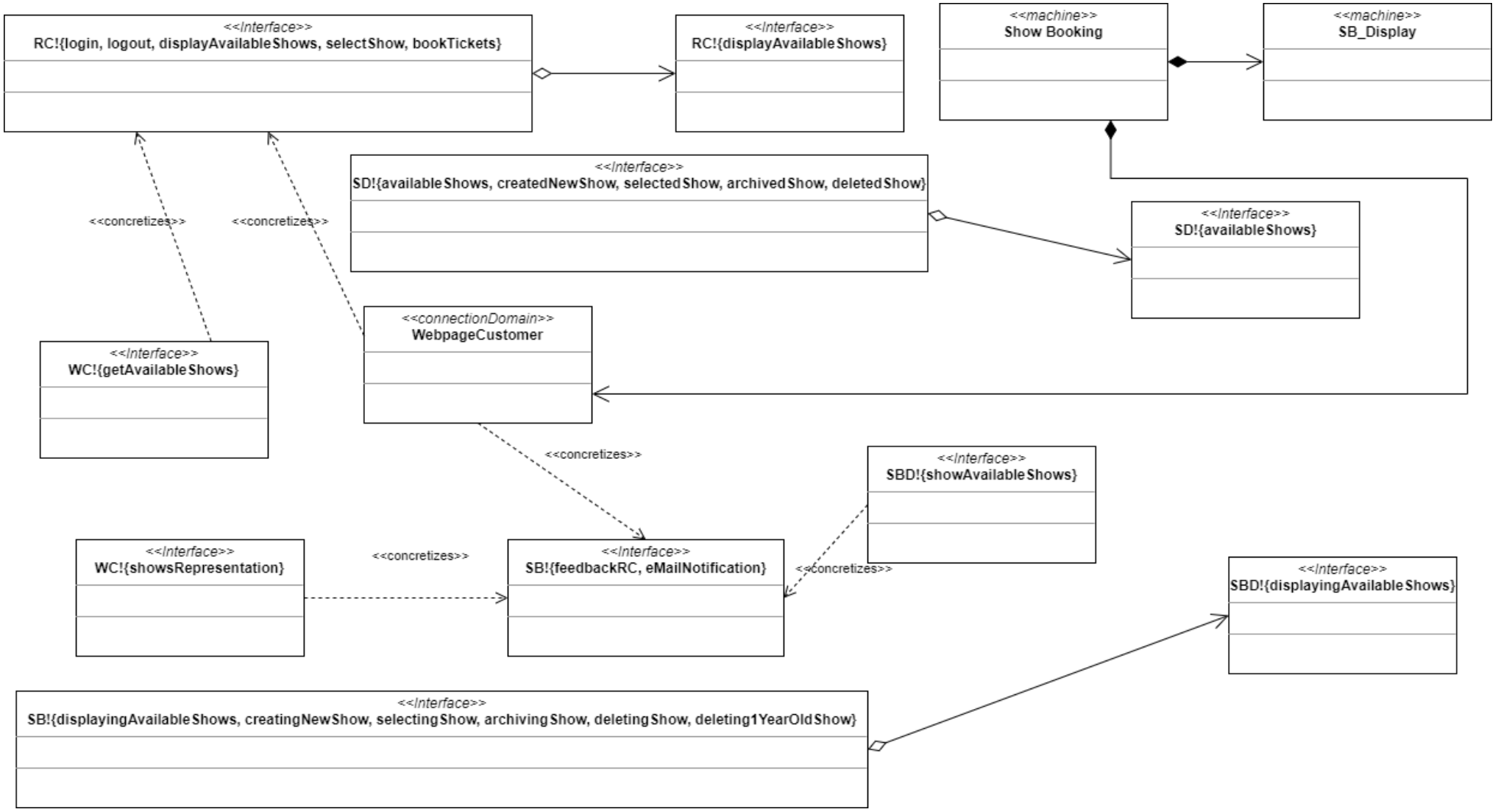


2. Customers should be able to display a list of future shows. This list should be sorted in ascending order of the time of the shows (next first) and each show should be clearly identifiable by an ID.

For this requirement we will use the problem frame **query II**

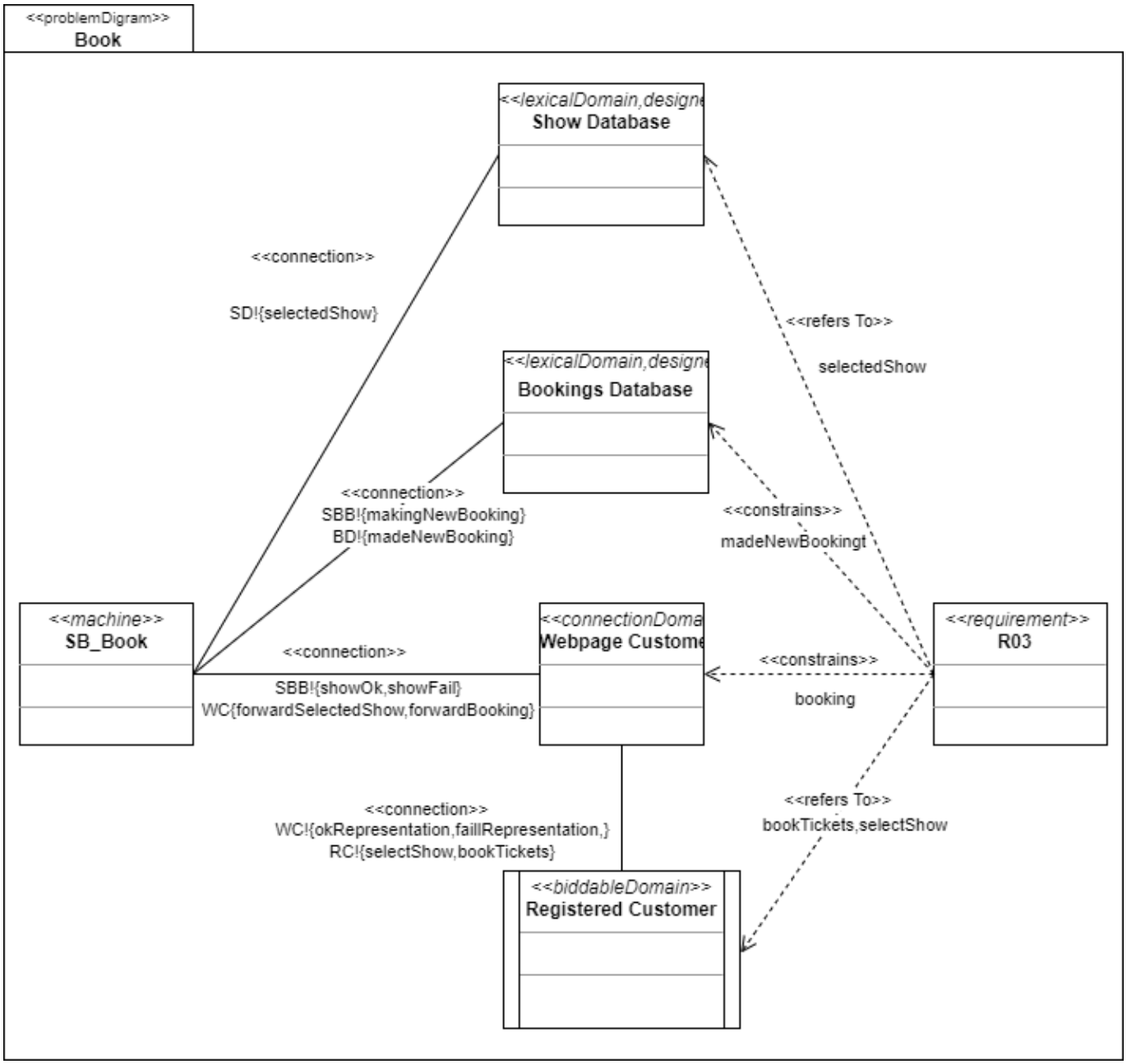


Concretize Interface(s):

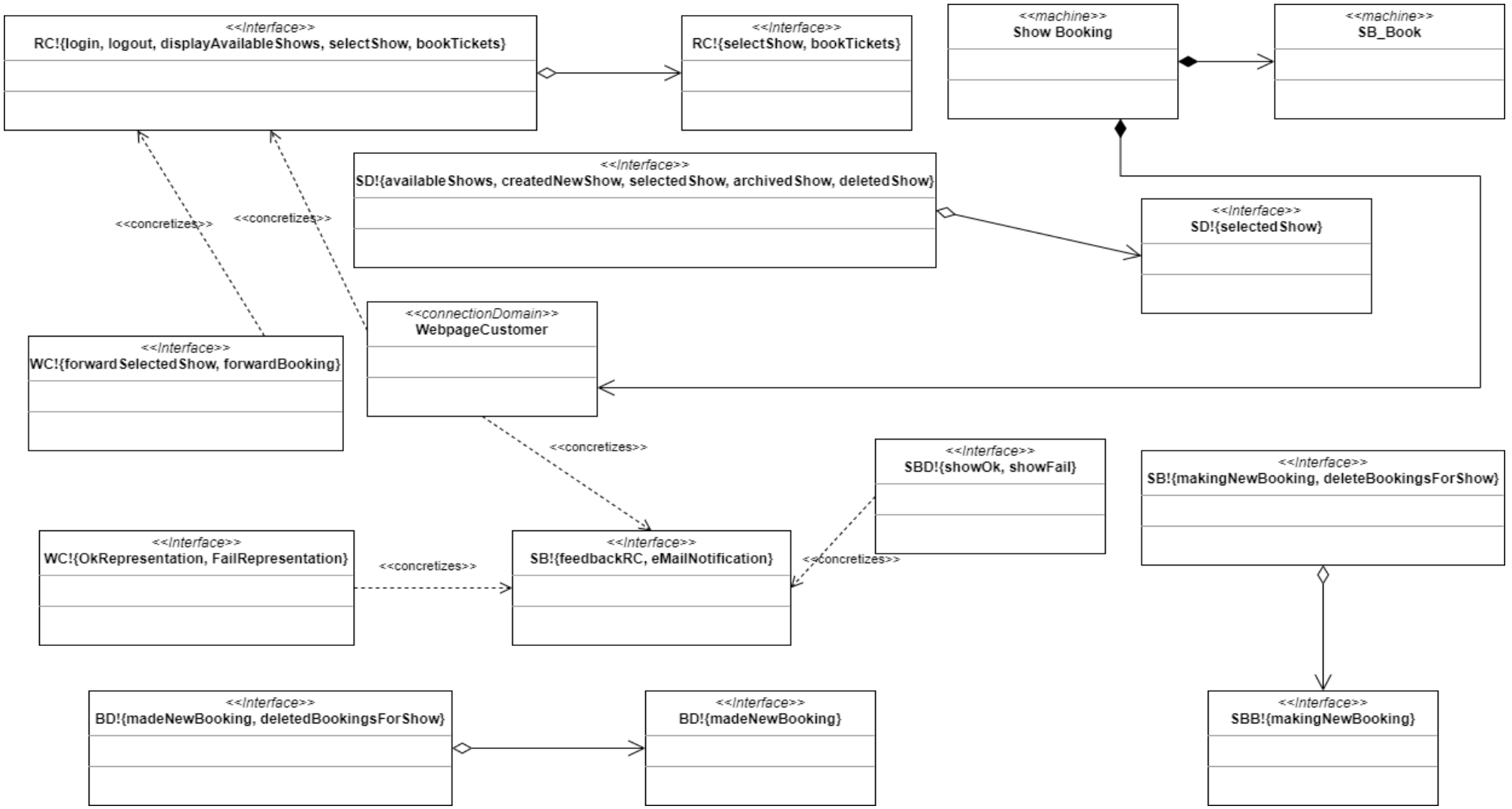


3. Registered customers can book tickets for a show by selecting it from the list and then specifying how many and which seats they want.This will only be possible to do no later than 15 minutes before the start of the show.

for this requirement we used the problem frame **update II**

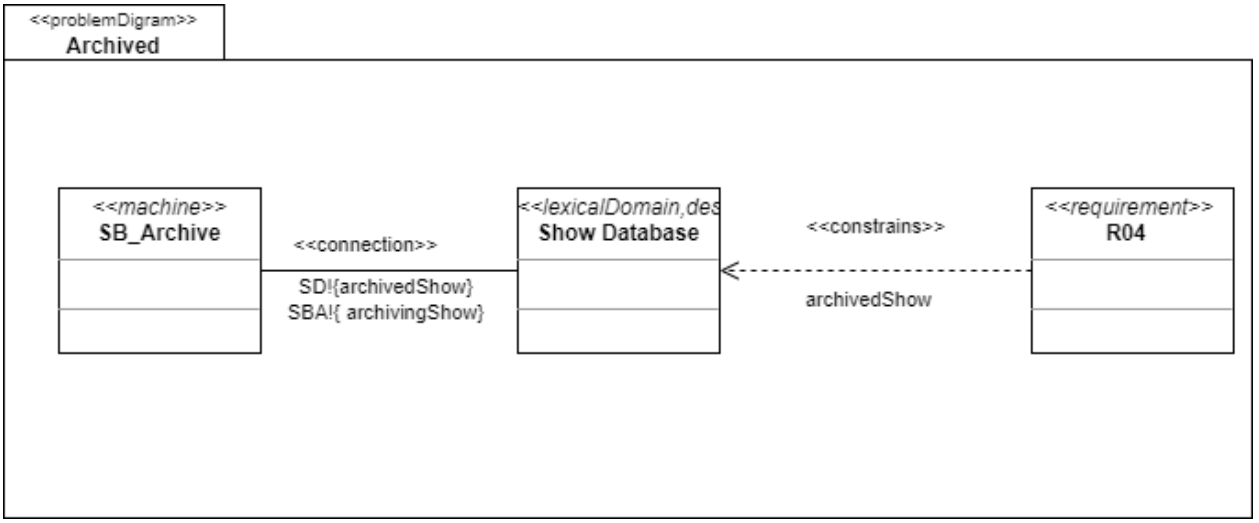


Concretize interface(s):

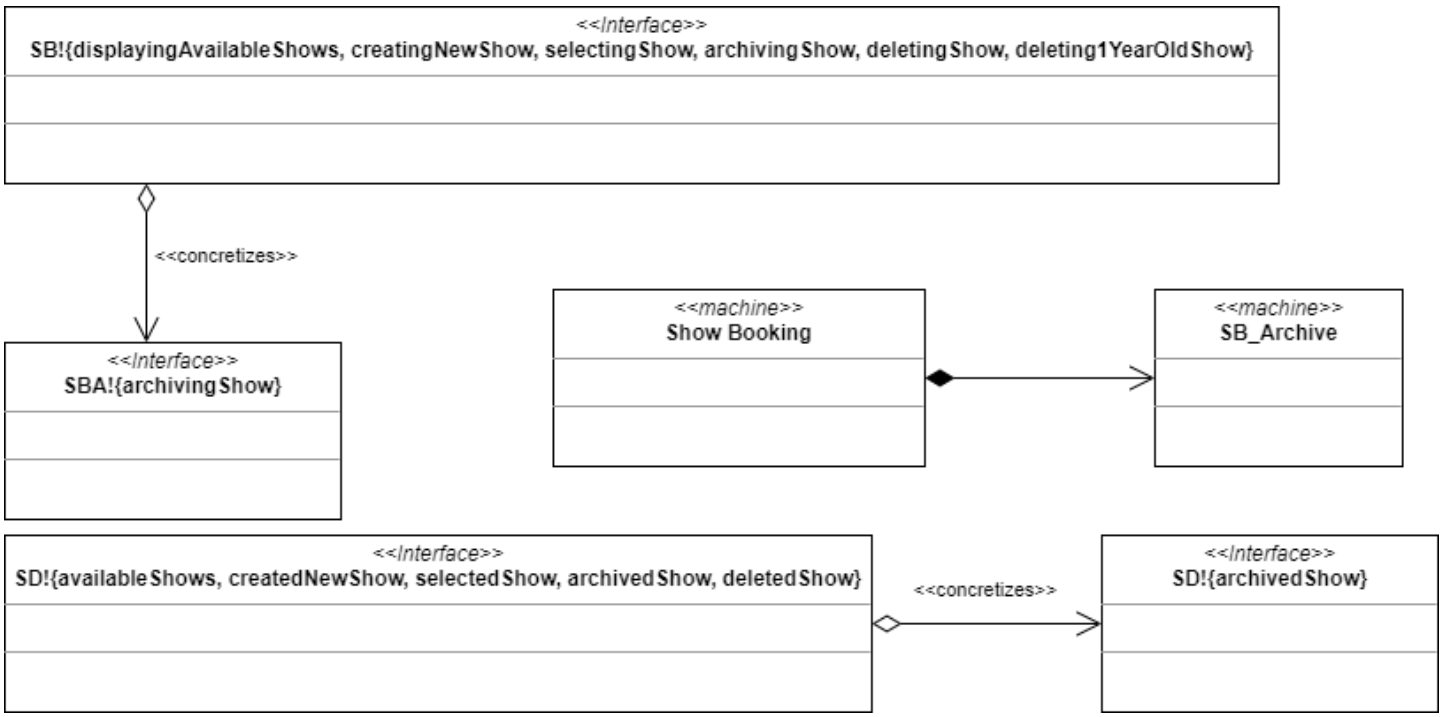


4. After a show has started, it should be marked as "archived" so that it is still available to the staff, but no longer appears in the list of shows.

for this requirement we used the problem frame **simple transformation**



concretize interface(s):



A3

Specifications

R01: In order to book a cinema ticket, customers must first register using a unique email address and a password.

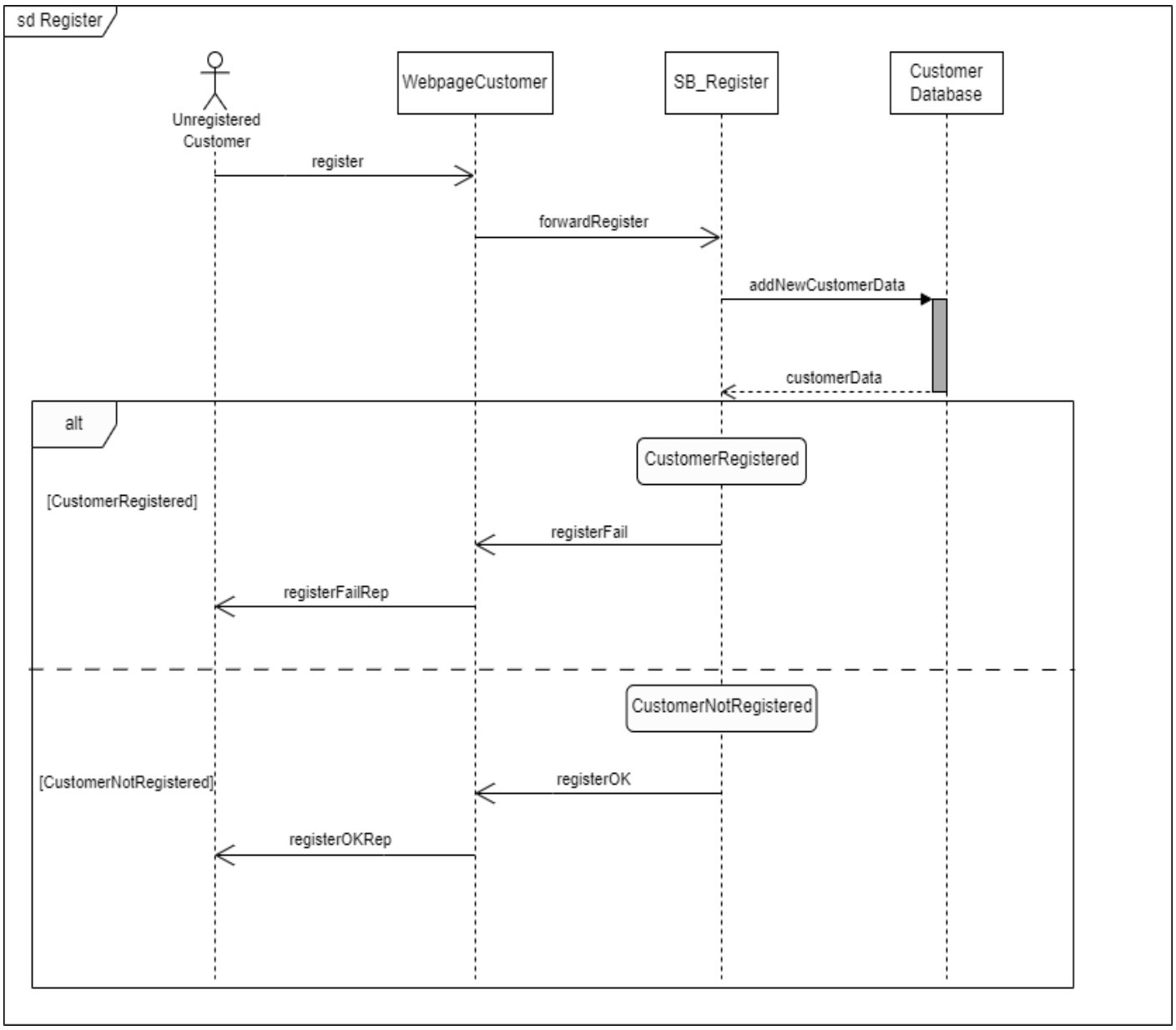
We Derive the following Specifications

WebpageCustomer (S01a) : When the webpage customer receives the command "register", then the command is forwarded to the machine with the command "forwardRegister". If the customer E-mail is used for the first time the results are received via the command "resgisterOk" and shown to the guest by "registerOKRep" else the results are received via the command "resgisterFail" (the customer E-mail is already registered) and shown to the guest by "registerFailRep".

SB_Register (S01b): When the machine receives the command "forwardRegister ,then the command is forwarded to the Customer Database with the command "addNewCustomerData" . The machine received the feedback via command "customerData". The results are returned via the command "registerOk" if the register was successful and "registerFail" otherwise.

Customer Database (S01c): After receiving the command "addNewCustomerData" from the machine. The data is added to our database and then the results are returned as the feedback "customerData".

Correctness condition: **(S01a)** $\wedge$ **(S01b)** $\wedge$ **(S01c)** $\wedge$ **(A01)** $\wedge$ **(A03)** $\Rightarrow$ **(R01)**


sd Register
Unregistered Customer
WebpageCustomer
SB_Register
Customer Database
register
forwardRegister
addNewCustomerData
customerData
alt
CustomerRegistered
[CustomerRegistered]
registerFail
registerFailRep
CustomerNotRegistered
[CustomerNotRegistered]
registerOK
registerOKRep

R02: Customers should be able to display a list of future shows. This list should be sorted in ascending order of the time of the shows (next first) and each show should be clearly identifiable by an ID.

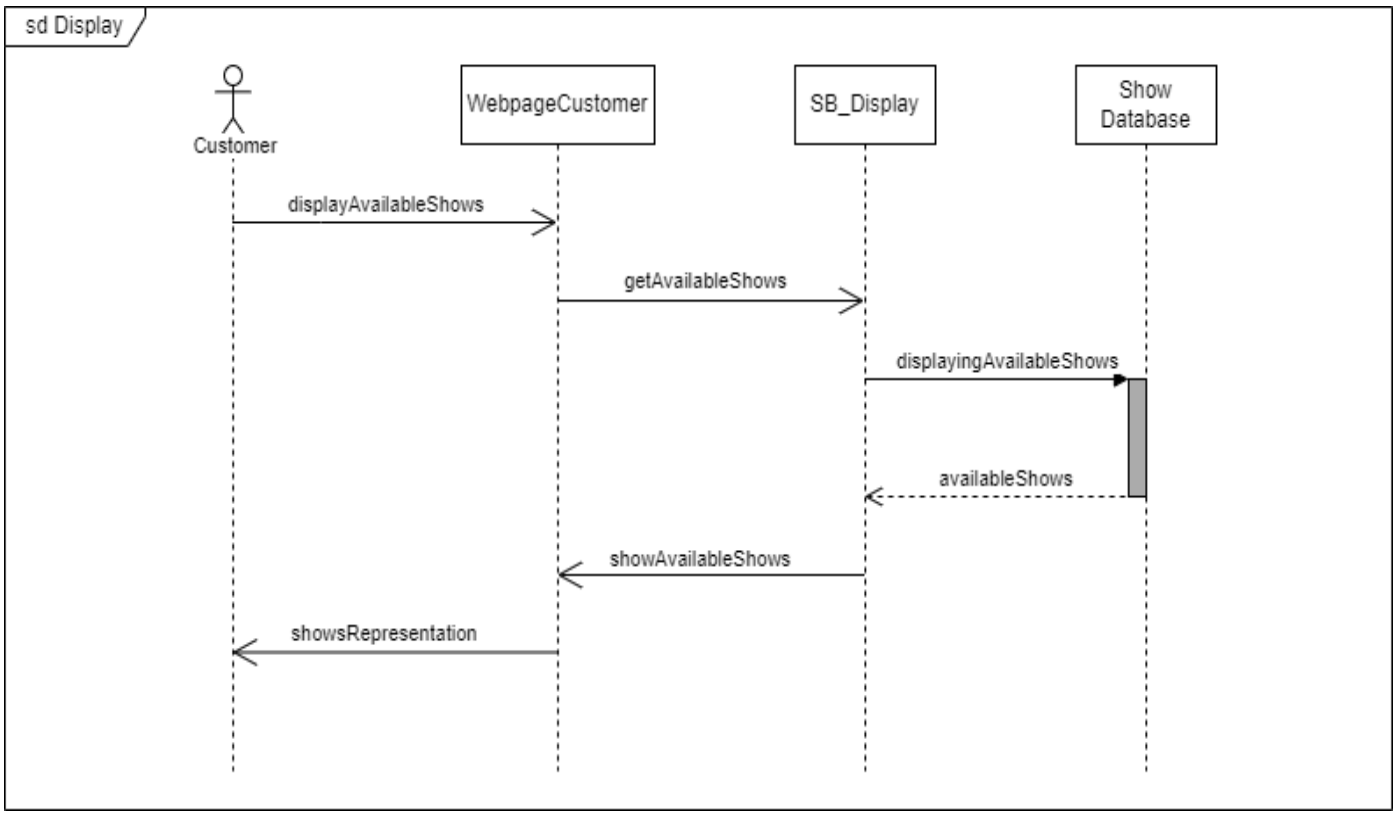
We Derive the following SpecificationsWebpageCustomer

WebpageCustomer (S02a) : When the webpage receives the command "displayAvailableShows", then the command is forwarded to the machine with the command "getAvailableShows". The results are received via the command "showAvailableShows" and shown to the customer by "showsRepresentation".

SB_Display (S02b): When the machine receives the command "getAvailableShows", the available shows are selected with the command "displayingAvailableShows" and received as the data "availableShows". The results are returned via the command "showAvailableShows".

Show Database (S02c): After receiving the command "displayingAvailableShows" the results are only the non-archived shows returned in a sorted manner as the data "availableShows".

Correctness condition: **(S02a)** ∧ **(S02b)** ∧ **(S02c)**∧ **(A02)** ∧ **(A03)** ∧ **(F01)** ⇒ **(R02)**



SShoww

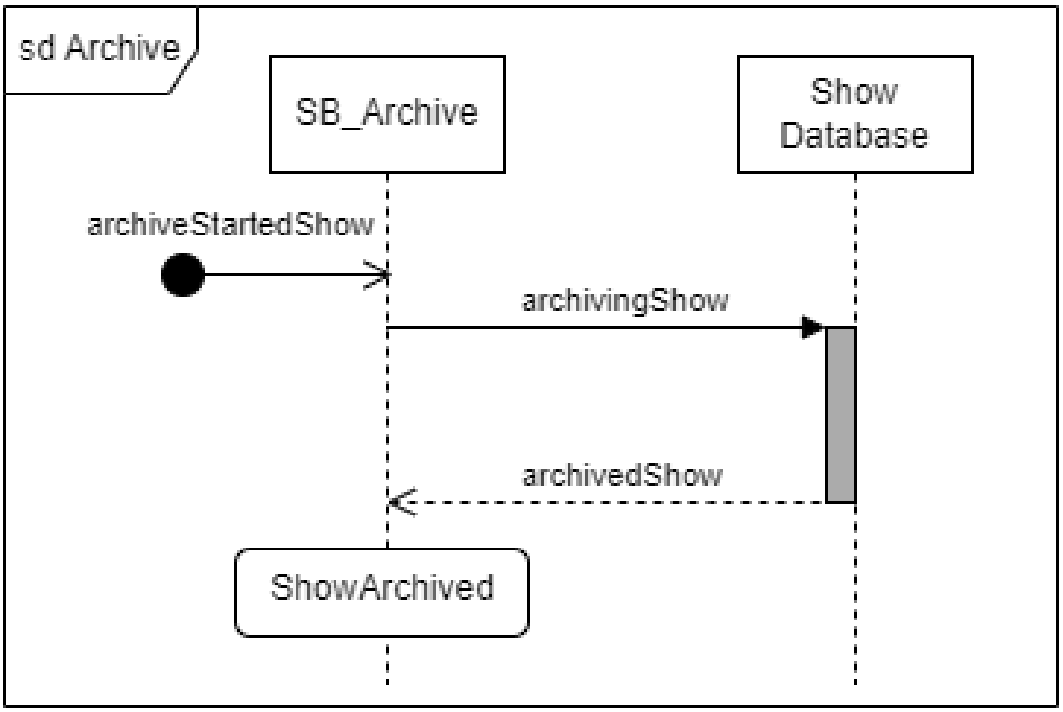
R04: After a show has started, it should be marked as "archived" so that it is still available to the staff, but no longer appears in the list of shows.

We Derive the following Specifications

SB_Archive (S04a): When the machine sends the command "archivingShow", the results are returned via the command "archivedShow".

Show Database (S04b): After receiving the command "archivingShow" the results are returned as the data "archivedShow".

Correctness condition: **(S04a)** ∧ **(S04b)** ∧ **(F01)** ⇒ **(R04)**



R03. Registered customers can book tickets for a show by selecting it from the list and then specifying how many and which seats they want.This will only be possible to do no later than 15 minutes before the start of the show.

Webpage Customer (S03a): The webpageCustomer receives the command "selectShow" by the customer which it forwards to the machine using the command "forwardSelectedShow", it then receives the command "ShowDetails" from the machine and forwards it to the customer using the command "ShowDetailsRep". When the Webpage receives the command "bookTickets", then it would forward it to the machine as "forwardBooking" . It would then receive from the machine the result via commands "showOk" , "showFail". The feedback is forwarded to the Guest via commands "okRepresentation" and "failRepresentation".

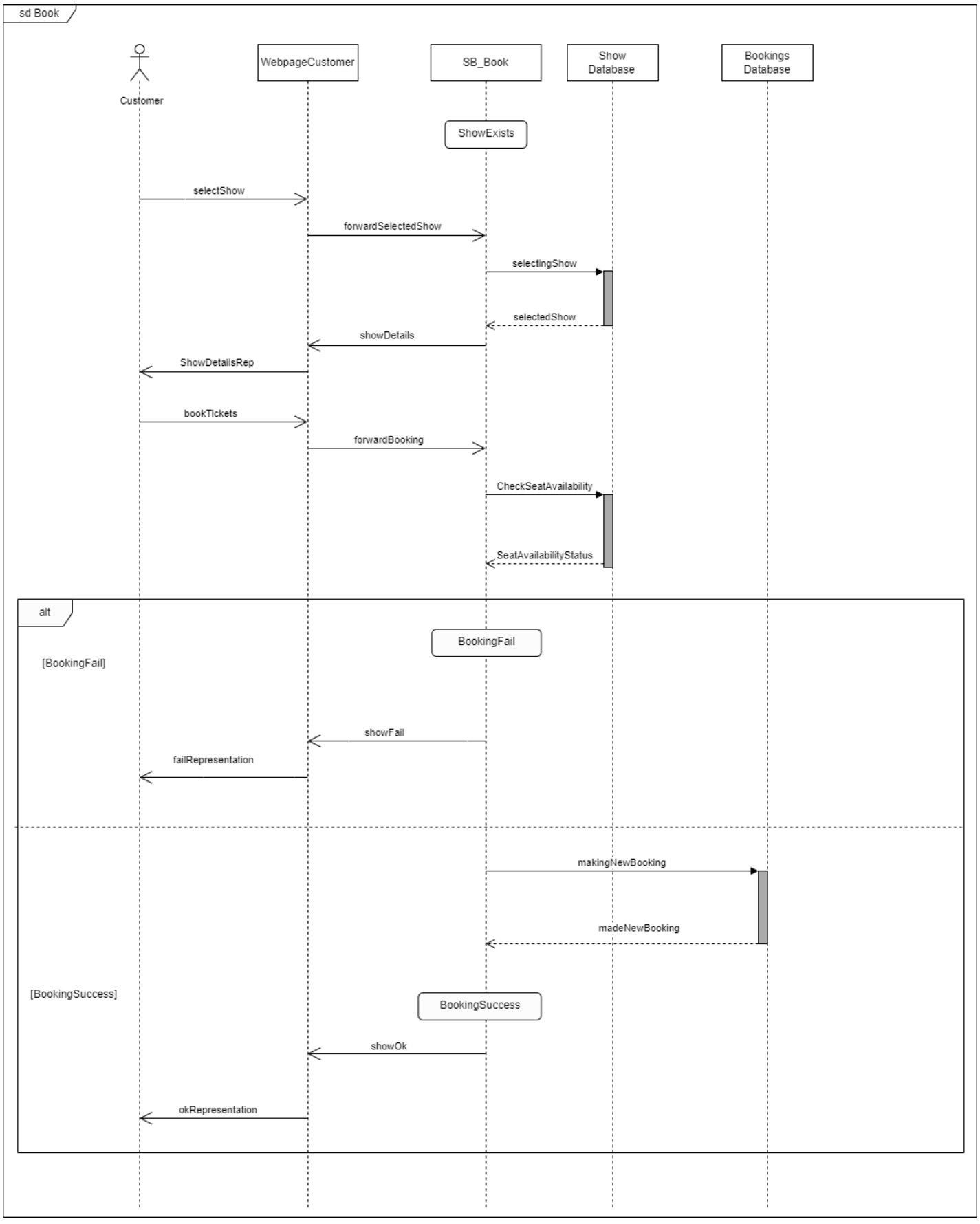
SB_Book (S03b): When the machine receives the command "forwardselectedShow" by the webpage customer it then retrieves the show from the database using "selectingShow" and the database would respond using the command "selectedShow". The machine would then respond to the webpage customer by sending it "ShowDetails". when the machine receives command "forwardbooking" , it would first check if it has been 15 minutes since the how started, then it would check if the capacity requirements are fulfilled from Bookings Database. the machine will receive the result from the database . if it is available the machine will send a new booking request to the Booking database via command "makingNewbooking" and the database will respond to it using "madeNewBooking". if all are done successfully it will send the acknowledge "showOk" to Webpage Customer otherwise it machine send "showFail"

Show Database (S03c): The show database receives the command "selectingShow" to return the wanted show it returns it using the command "selectedShow".

Booking Database (S03d): when receiving "makingNewBooking"' it would make the booking and send a message back to the machine "madeNewBooking" .

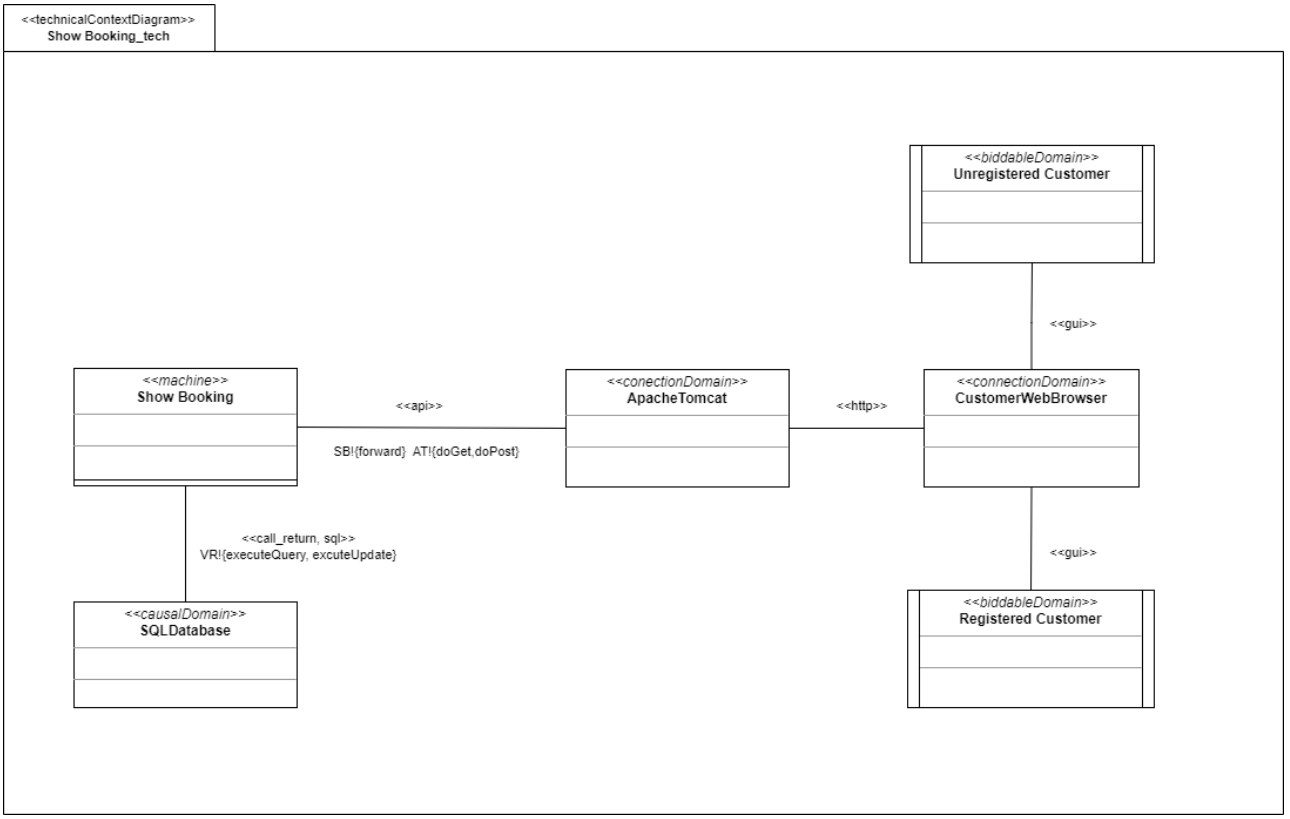
Correctness condition:

(S03a) $\wedge$ **(S03b)** $\wedge$ **(S03c)** $\wedge$ **(S03d)** $\wedge$ **(A03)** $\wedge$ **(A04)** $\wedge$ **(F01)** $\wedge$ **(F02)** $\Rightarrow$ **(R03)**

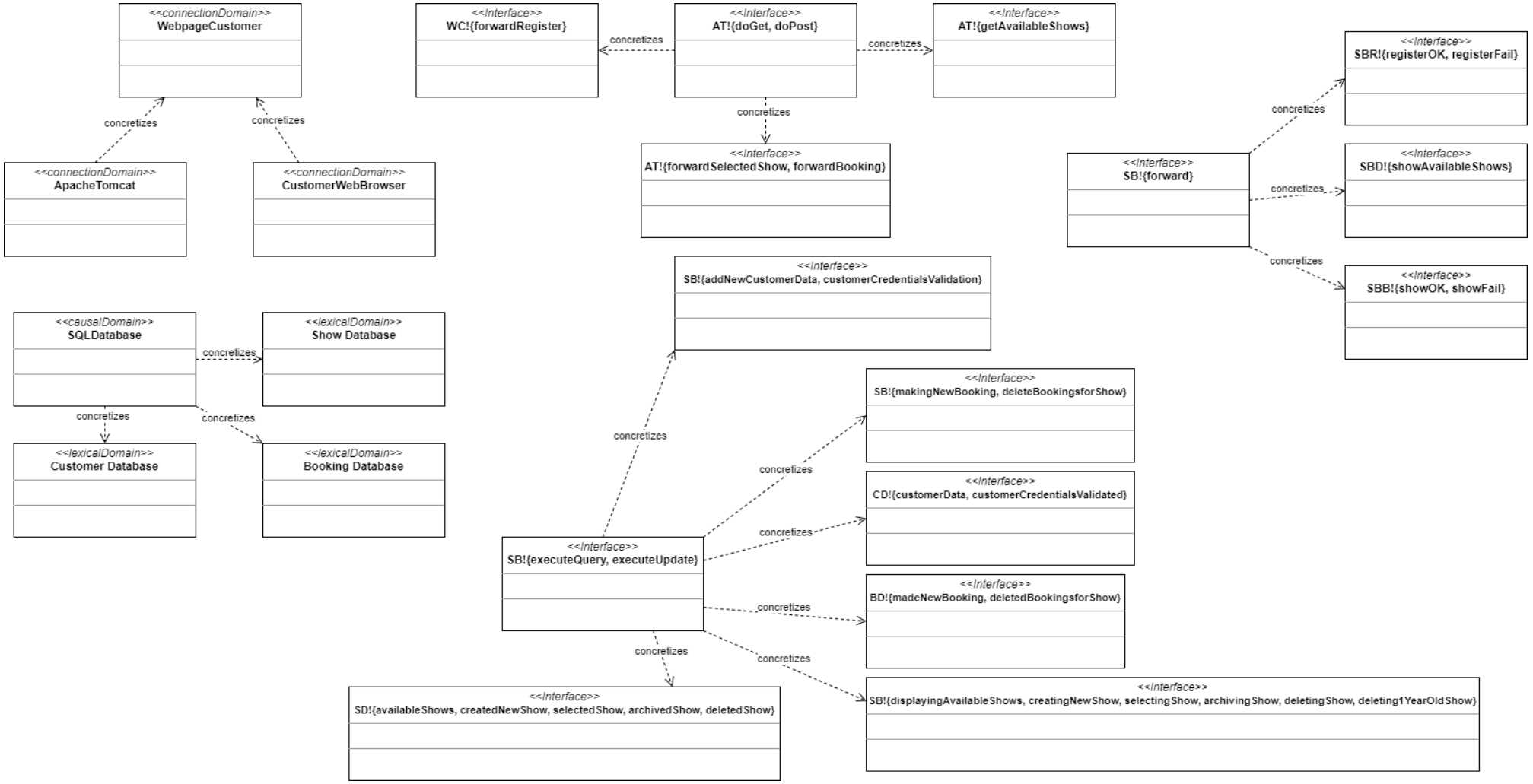

sd Book
Customer
WebpageCustomer
SB_Book
Show Database
Bookings Database
ShowExists
selectShow
forwardSelectedShow
selectingShow
selectedShow
showDetails
ShowDetailsRep
bookTickets
forwardBooking
CheckSeatAvailability
SeatAvailabilityStatus
alt
BookingFail
[BookingFail]
showFail
failRepresentation
makingNewBooking
madeNewBooking
[BookingSuccess]
BookingSuccess
showOk
okRepresentation

A4

Technical Context Diagram



Mapping Diagram



A5

Class Model for Customer Database

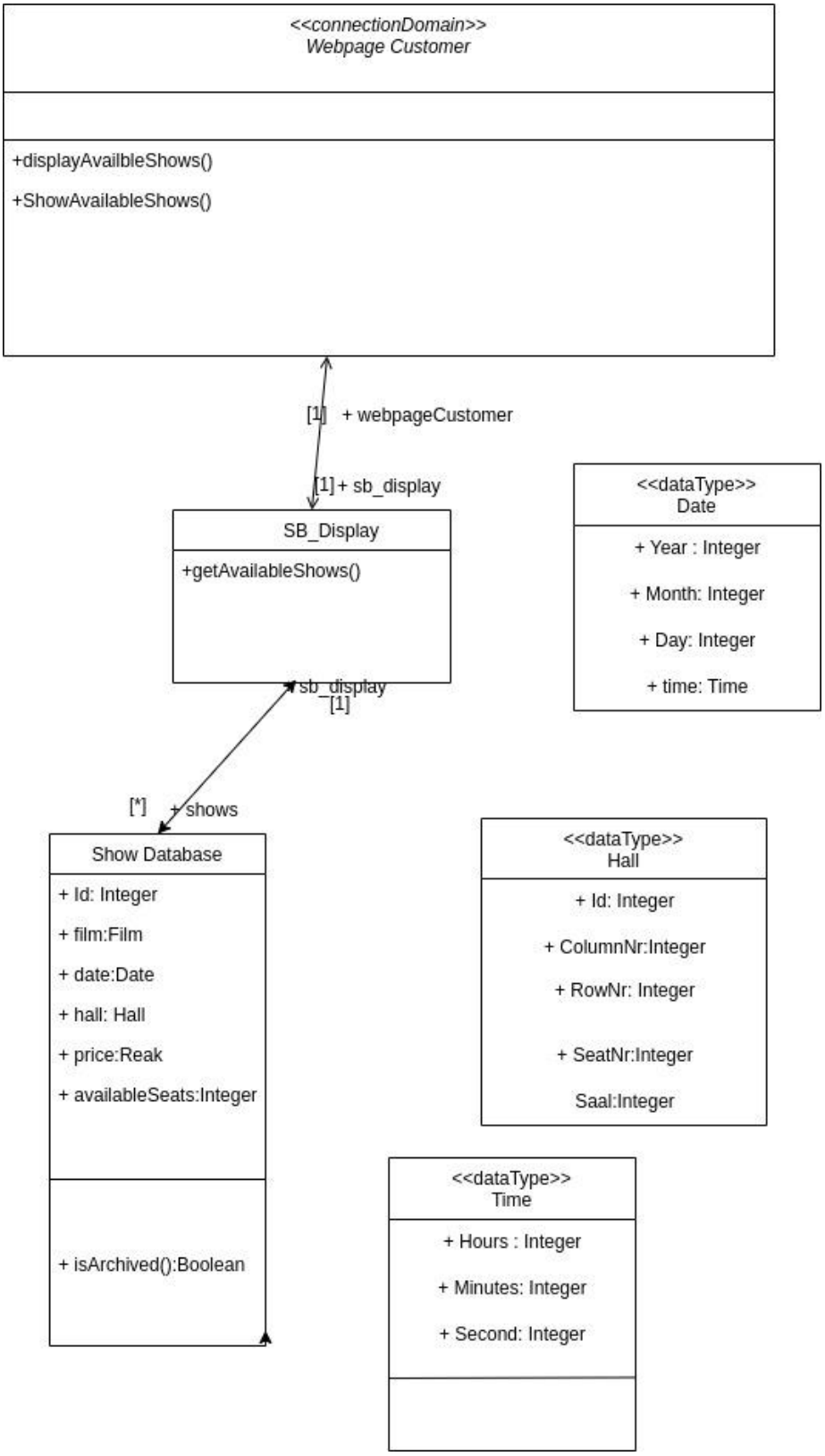
OCL

Name: forwardRegister

Description: Execute the registration request of the customer

```
Context SB::Register::forwardRegister (firstName:String , lastName:String , email:String ,
password:String ,bitdaate:Date , address:AddressData)

Pre:true

Post:if customers->size() - customers@pre->size()=1 and

Not customers@pre->exist( c: Customer |

c.lastName= lastName and  c.firstName = firstName and

c.Password = Password and c.emal = email )

and customers-> one ( c: Customer |

c.lastName= lastName and  c.firstName = firstName and

c.Password = Password and c.emal = emai )

Then Else WepWebpageCustomer ^ registerOK()

Else WebpageCustomer ^ registerFail()

endif
```

Class Model for Show Database



OCL

Name: getAvailableShows

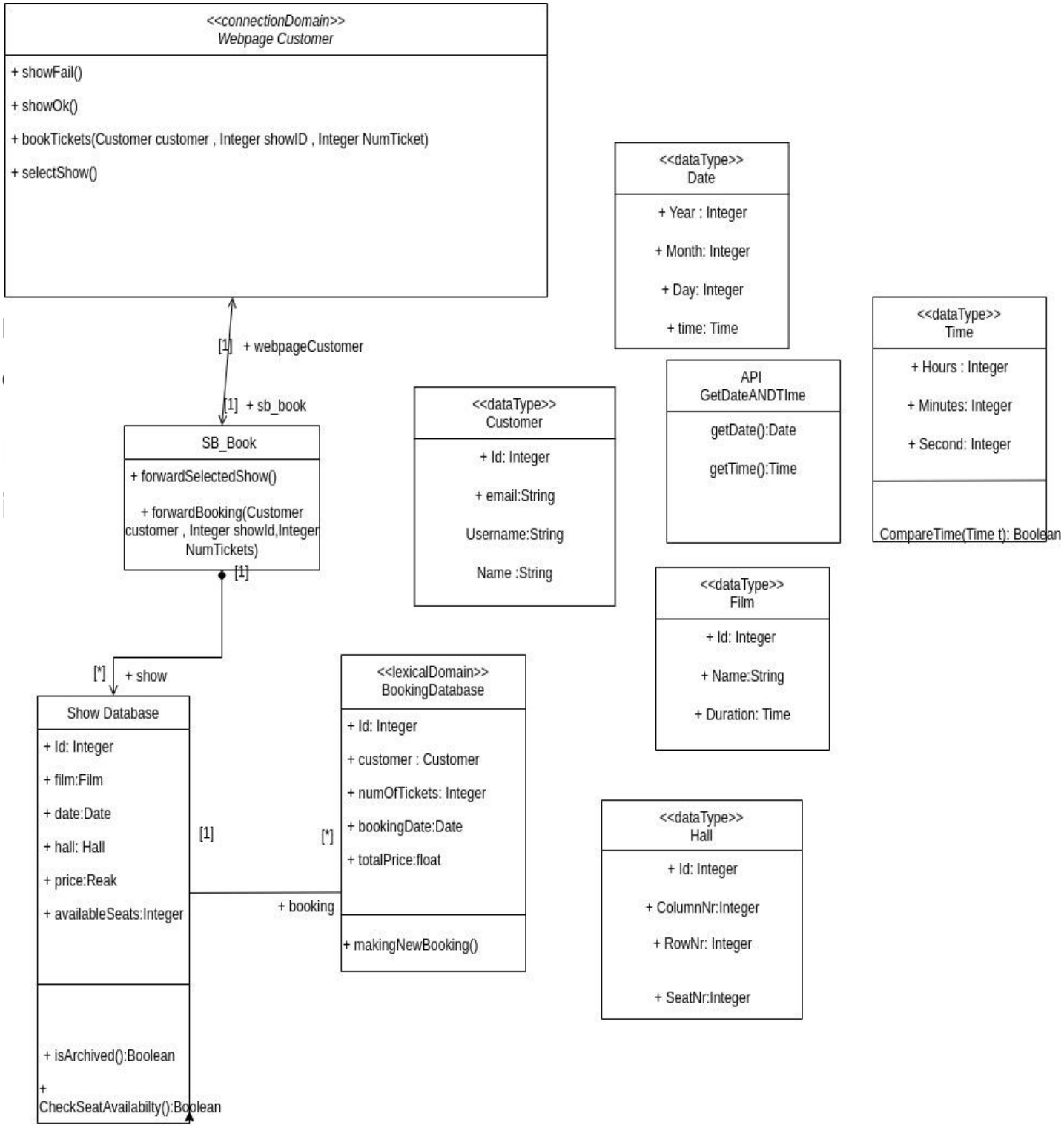
Description: Select all non archived shows and returned it to customers

Remark: isArchived (): return Boolean if the show is archived or not

```
Context:SB_Display::getAvailableShows()
Pre:true
Post:let res:OrderdSet(ShowDatabase)+
shows->select (show:ShowDatabase |
show.isArchived = False ) asOrderedSet()
in
webPageCustomer ^showAvaailbleShows(res)
```

```
Context:shows
Inv:shows.allInstance()->isUnique(id)
```

Class Model


<<connectionDomain>>
Webpage Customer
+ showFail()
+ showOk()
+ bookTickets(Customer customer , Integer showID , Integer NumTicket)
+ selectShow()
[1] + webpageCustomer
[1] + sb_book
SB_Book
+ forwardSelectedShow()
+ forwardBooking(Customer customer , Integer showId,Integer NumTickets)
[1]
[*] + show
Show Database
+ Id: Integer
+ film:Film
+ date:Date
+ hall: Hall
+ price:Reak
+ availableSeats:Integer
+ isArchived():Boolean
+ CheckSeatAvailabilty():Boolean
[1]
[*]
+ booking
<<lexicalDomain>>
BookingDatabase
+ Id: Integer
+ customer : Customer
+ numOfTickets: Integer
+ bookingDate:Date
+ totalPrice:float
+ makingNewBooking()
<<dataType>>
Customer
+ Id: Integer
+ email:String
Username:String
Name :String
<<dataType>>
Date
+ Year : Integer
+ Month: Integer
+ Day: Integer
+ time: Time
API
GetDateANDTIme
getDate():Date
getTime():Time
<<dataType>>
Time
+ Hours : Integer
+ Minutes: Integer
+ Second: Integer
CompareTime(Time t): Boolean
<<dataType>>
Film
+ Id: Integer
+ Name:String
+ Duration: Time
<<dataType>>
Hall
+ Id: Integer
+ ColumnNr:Integer
+ RowNr: Integer
+ SeatNr:Integer

OCL

Name: forwardBooking

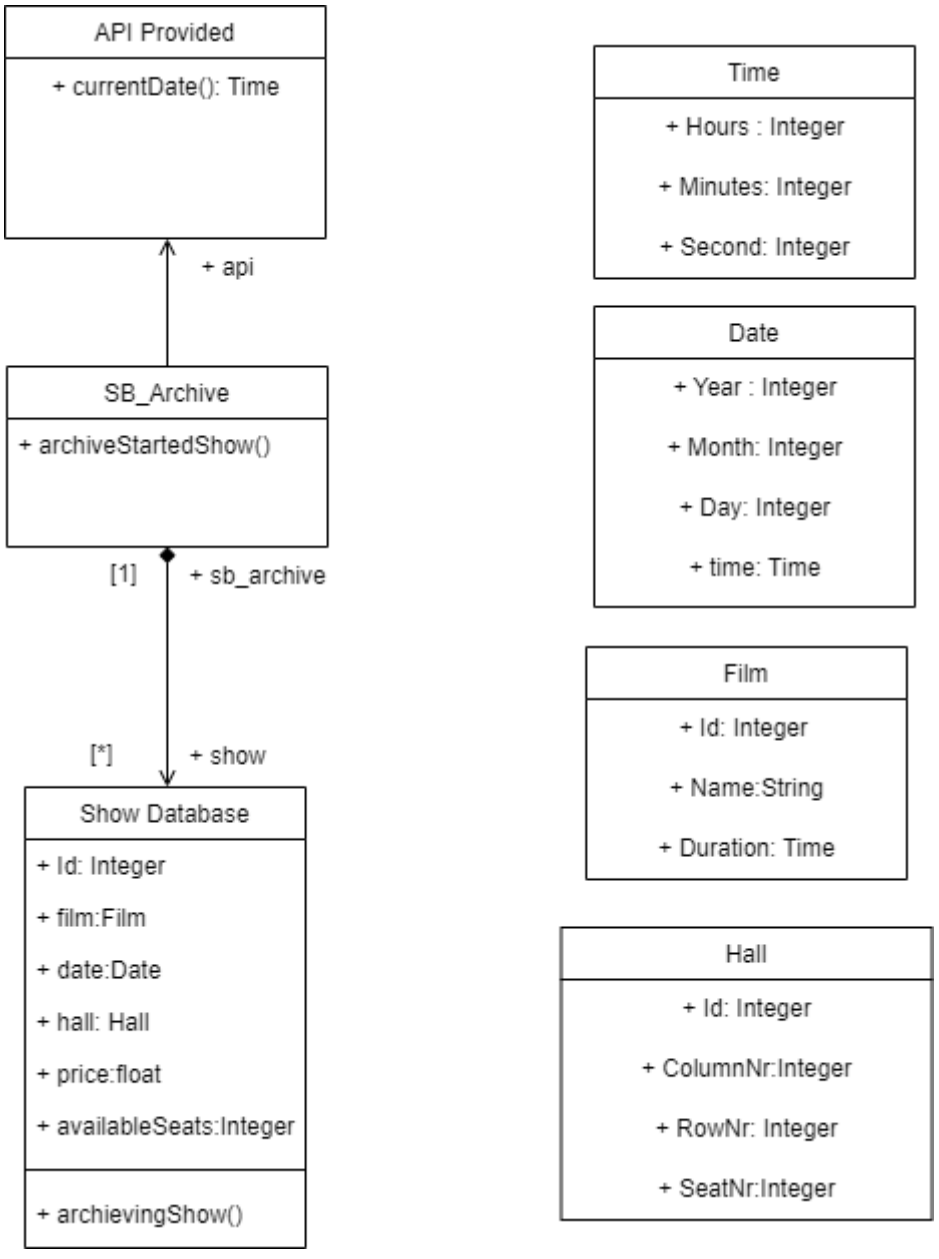
Description: forward customer booking show request

```
Context SB_Book:: forwardBooking (numTickets:Integer , customer:Customer ,showId:Integer )
Pre:shows -> one (s:ShowDatabase | s.id = showID )
Post: let
resultShow:Show = shows ->any (sh:ShowDatabase | sh.id = showId )
in
if numberTickets <= resultShow @pre.availableSeats And resultShow.time -api.getTime() >= 15
then  resultShow.booking-> one ( book :bookingDatabase |
book.customer = customer and
book.numOfTickets = numTickets and
book.bookingDate = app.getDate() )and
resultShow.booking ->size() = resultShow.booking@pre ->size()+1
and wepageGuest ^ showOk()
else wepPageGuest^showFai() endIf
```

```
Context :Booking
Inv: Booking.allInstances()->isUnique(id)
```

Context: shows

Inv: shows.allInstances()->isUnique(id)



OCL

Name: archiveStartedShow

```
Context: SB_Archive::archiveStartedShow(showId:Integer)
Pre : true
Post : let
resultShow:Show=shows->any(sh:Show |sh.id=showId)
in
if resultShow.date >=api.currentdate() then
resultShow.isArchieved() =true
else resultShow.isArchieved()=false
endif
```

A6

Life Cycle

$LC_{\text{Unregistered Customer}}$ = (Register)*

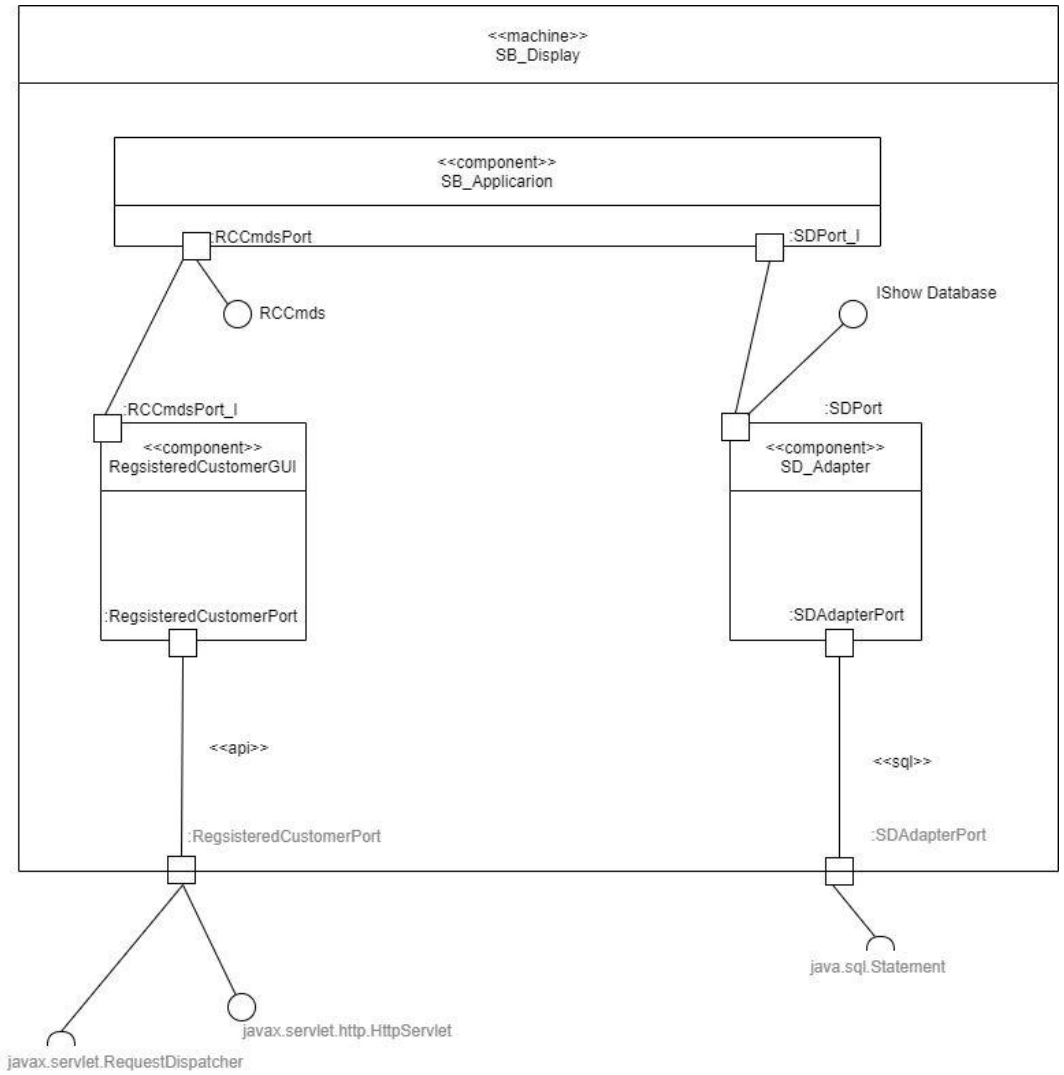
$LC_{\text{Registered Customer}}$ = (Login;[Display;[Select^{+};[Book*]]]*;Logout)*

$LC_{\text{Show Booking}} = (||_{i=1}^{n} LC_{\text{Unregistered Customer}}) \,||\, (||_{j=1}^{m} LC_{\text{Registered Customer}}) \,||\, \text{Archive}^*$

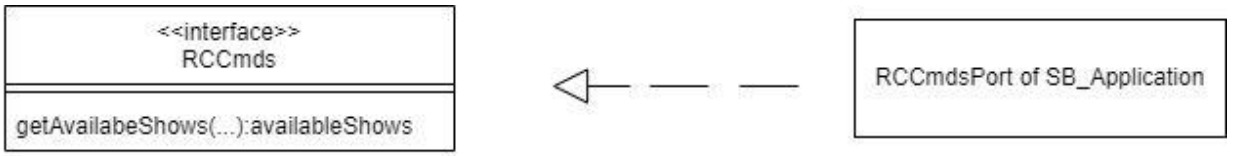
D1

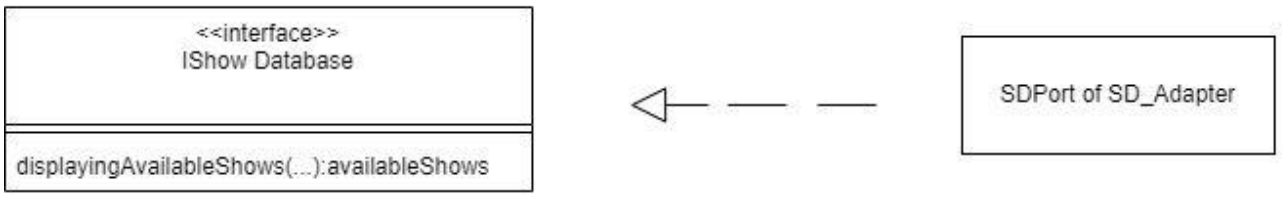
Subproblem architectures

SB_Display fits to query(2). Instantiated architectural pattern for SB_Display

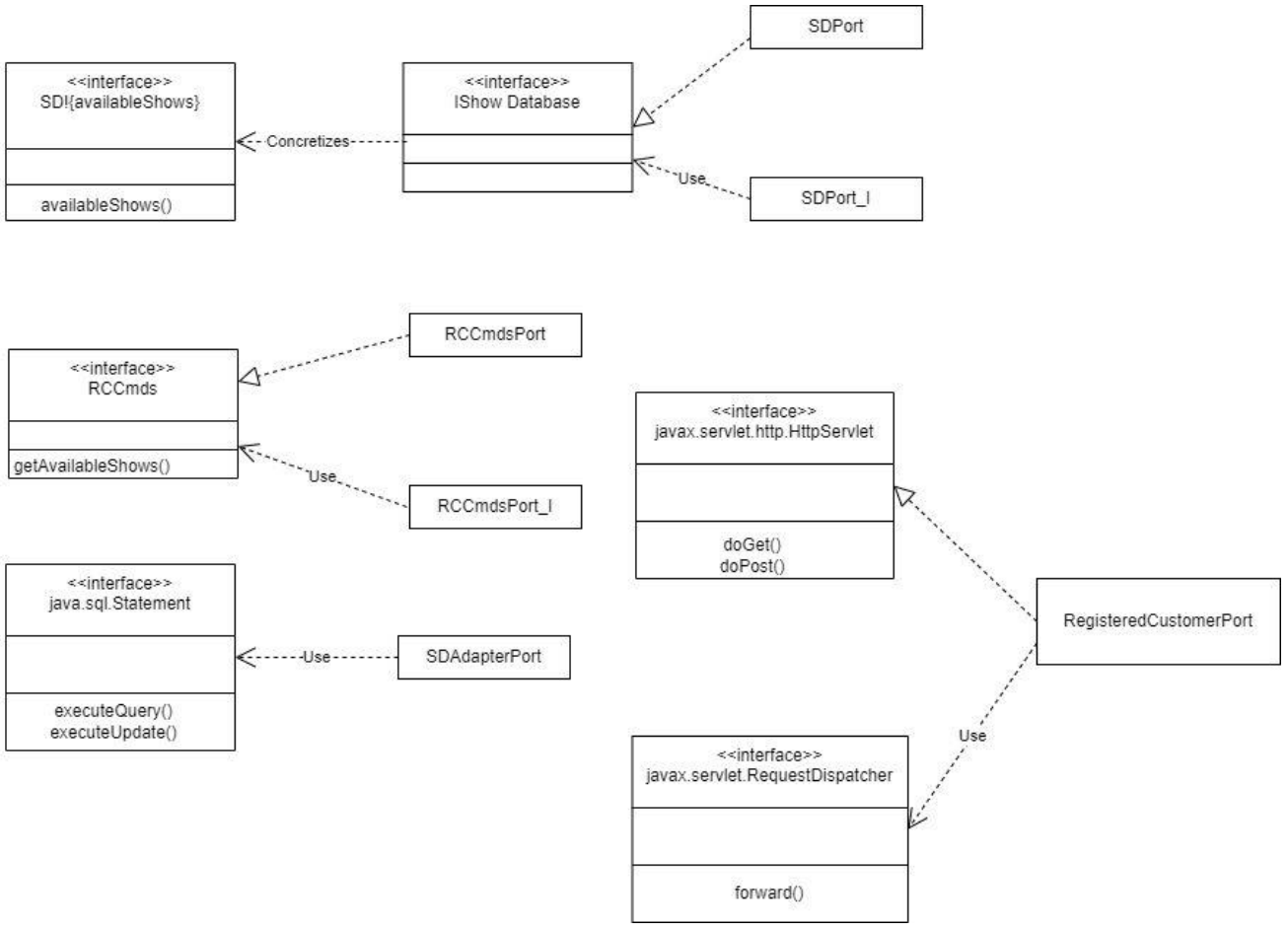


Internal interfaces in SD_Display

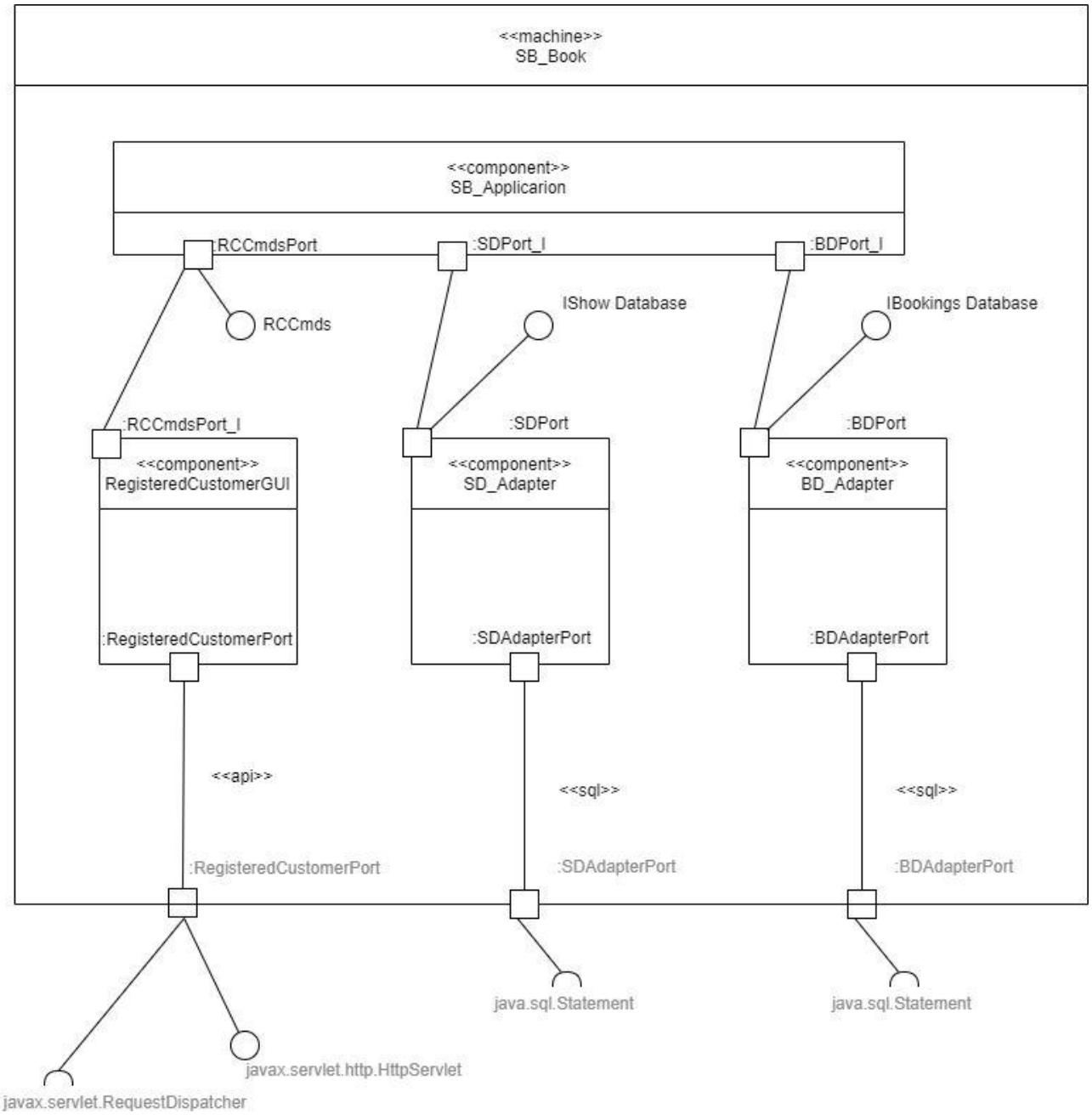




Port types and interface relations for SD_Display



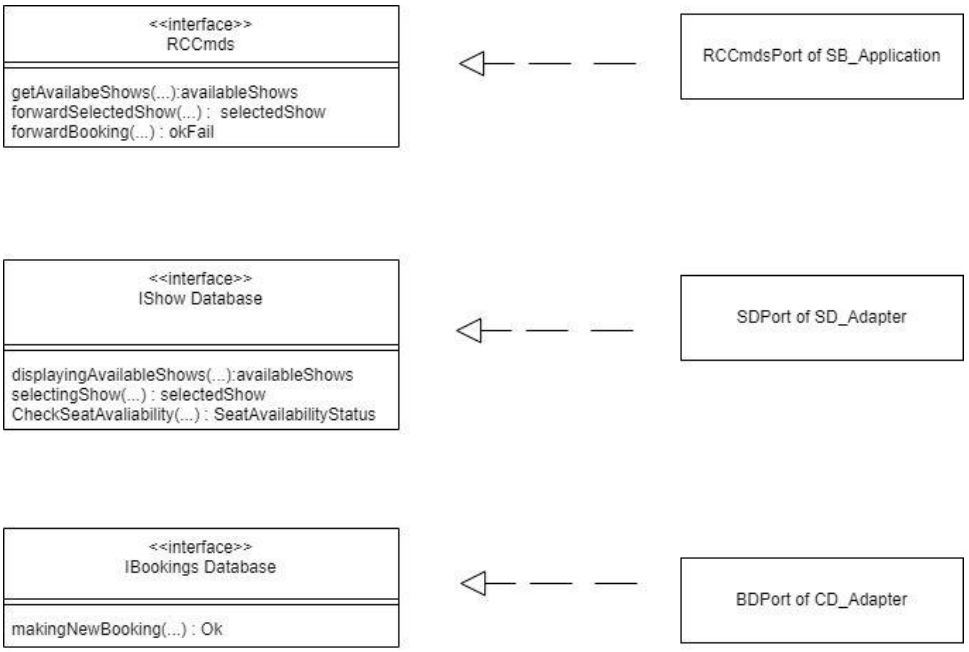
SB_Book fits to update(2). Instantiated architectural pattern for SB_Book



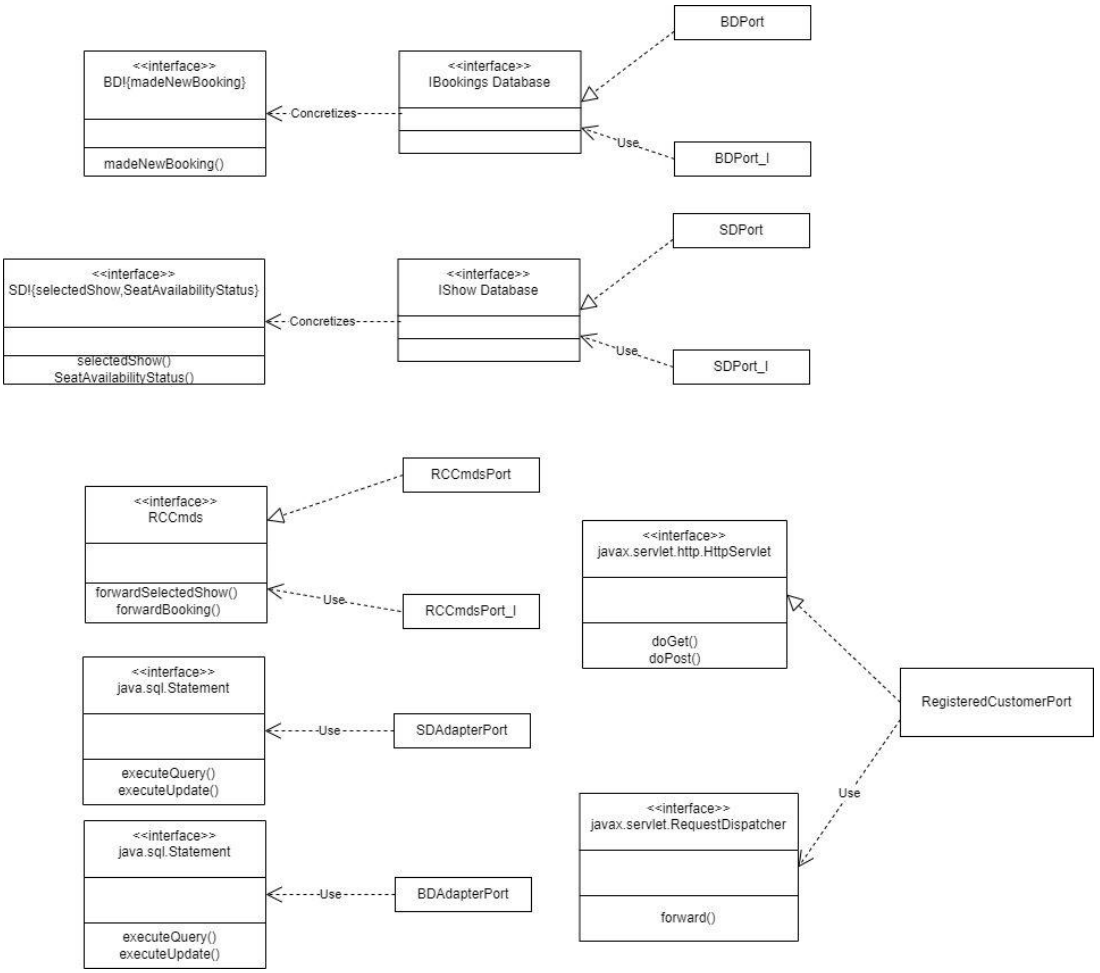
Internal interfaces in SB_Book

RCCmds from SD_Display is extended as follows

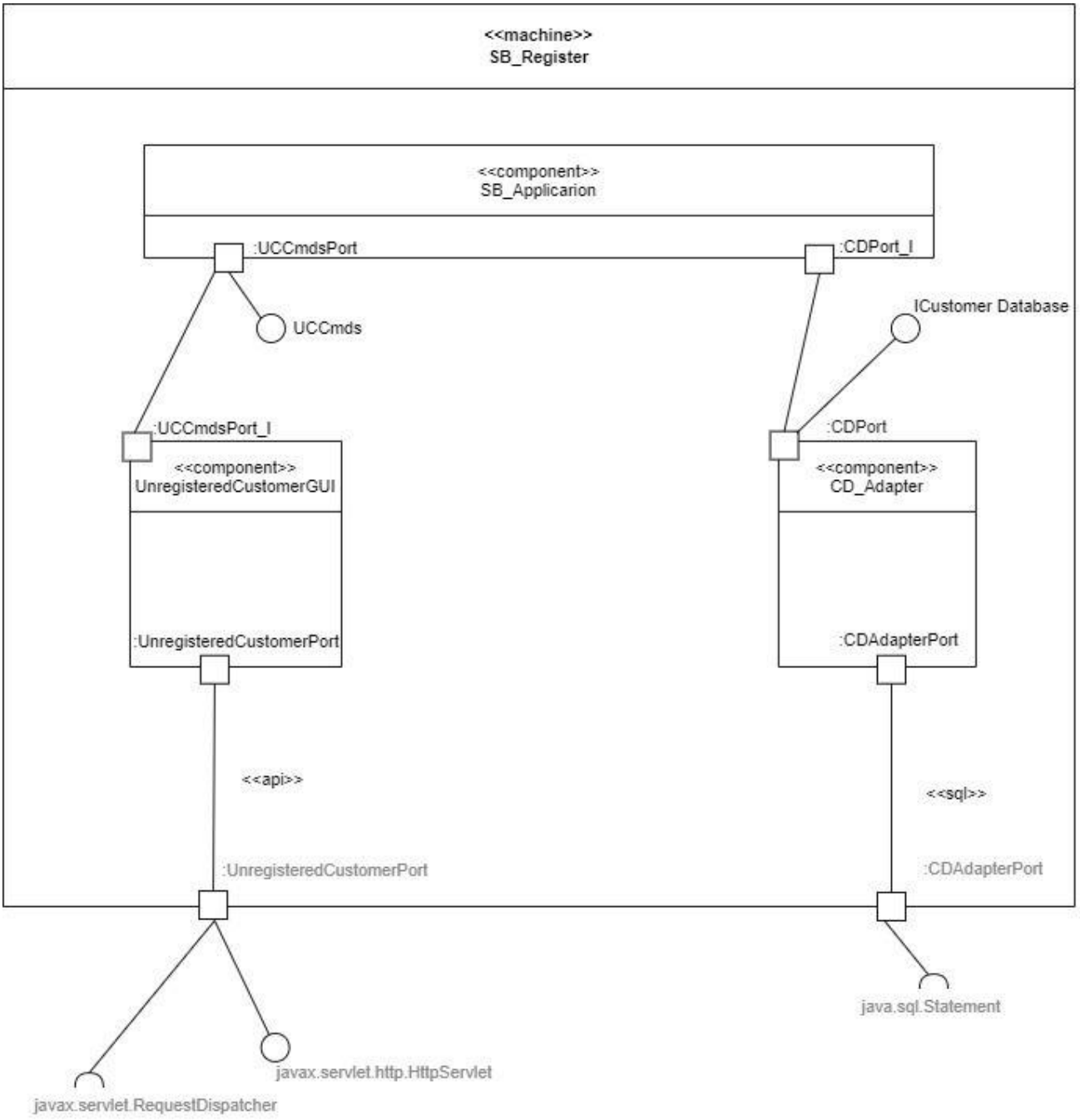
and IShow from SD_Display is extended as follows



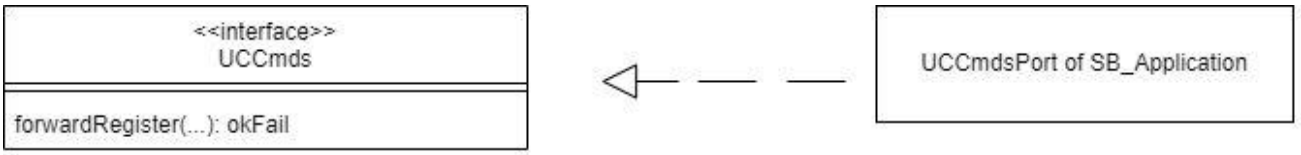
Port types and interface relations for SB_Book

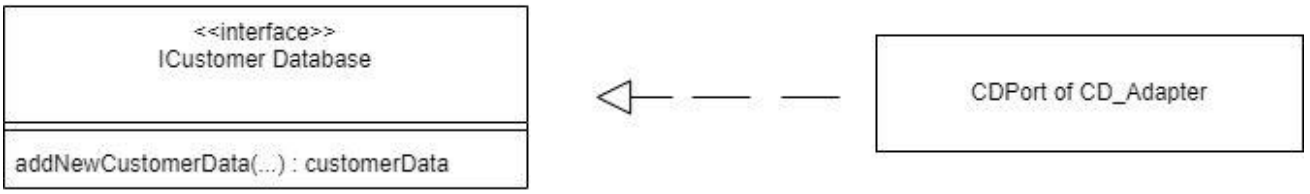


SB_Register fits to update (2). Instantiated architectural pattern for SB_Register

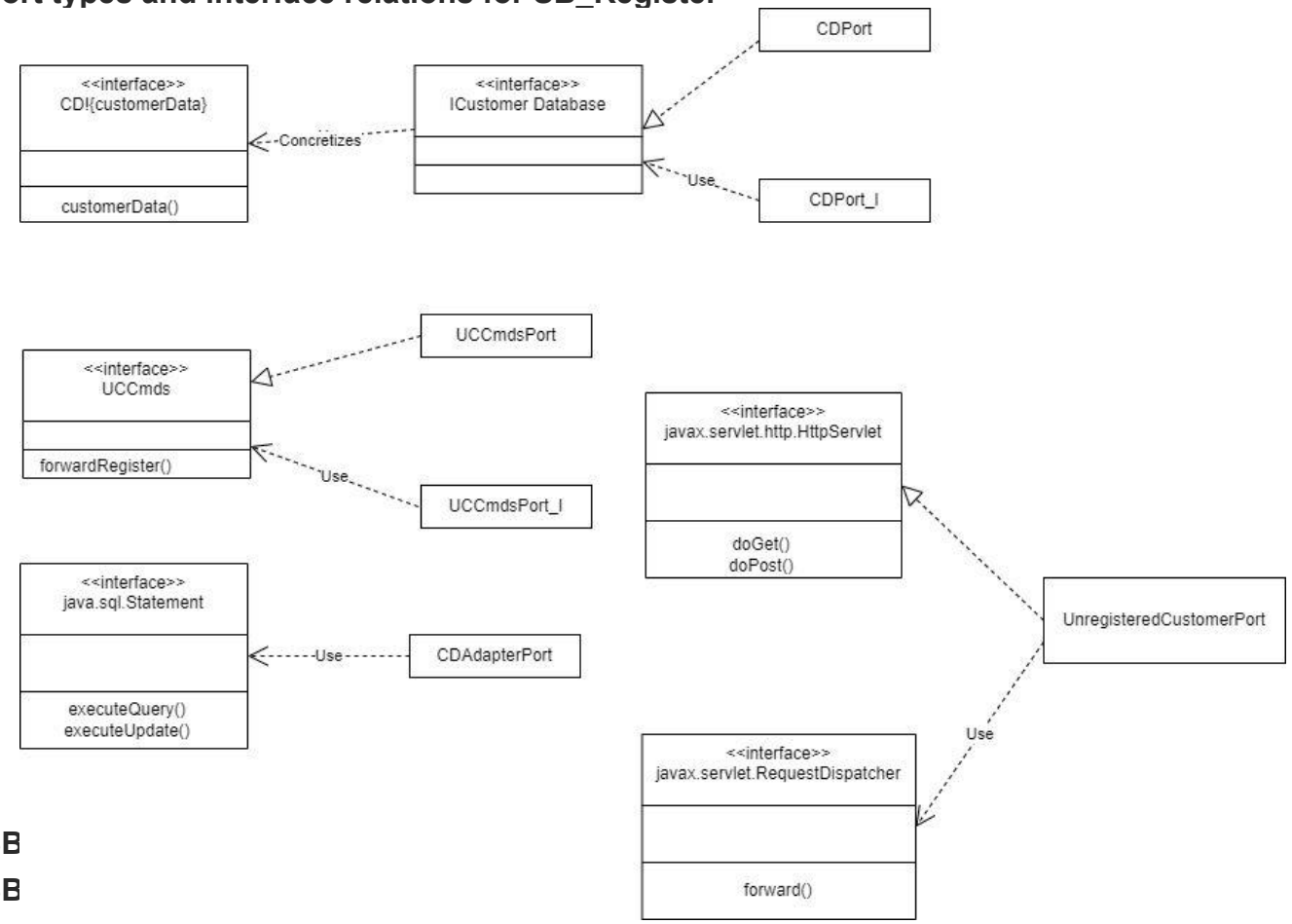


Internal interfaces in SB_Register



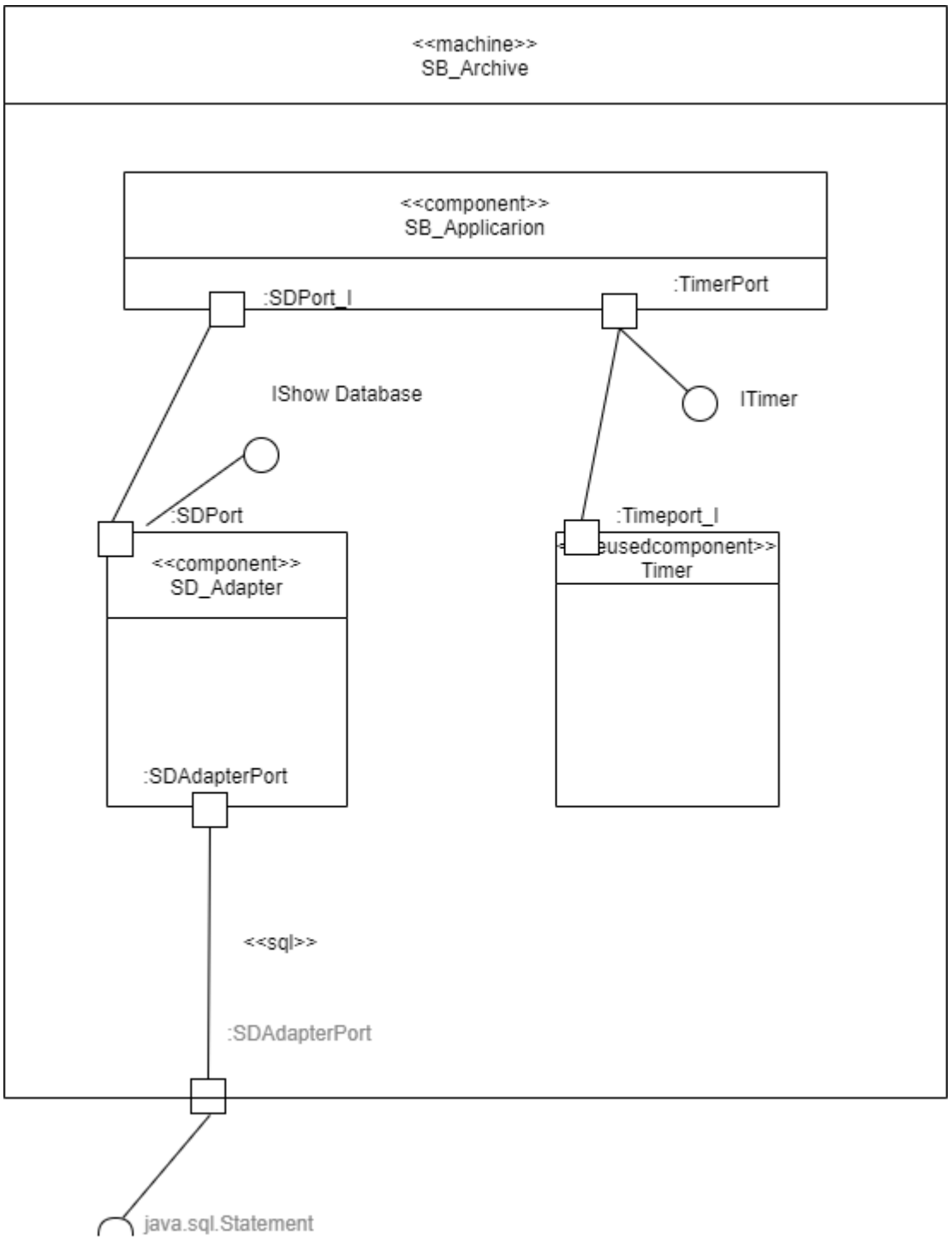


Port types and interface relations for SB_Register

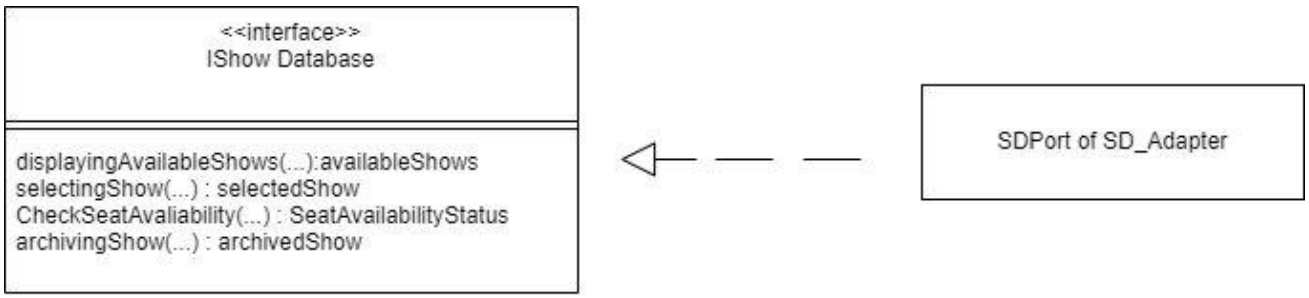


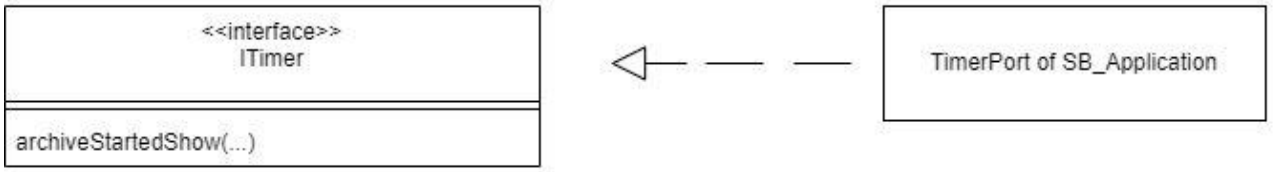
SB

SB

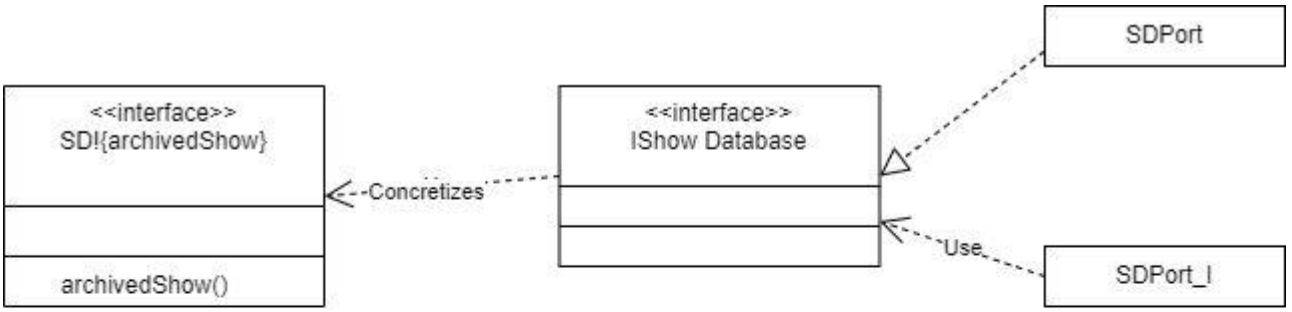


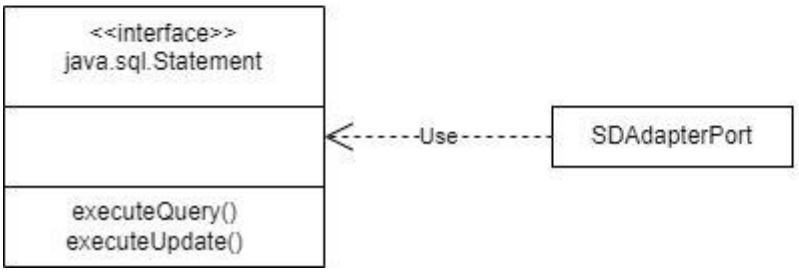
Internal interfaces in SB_Archive



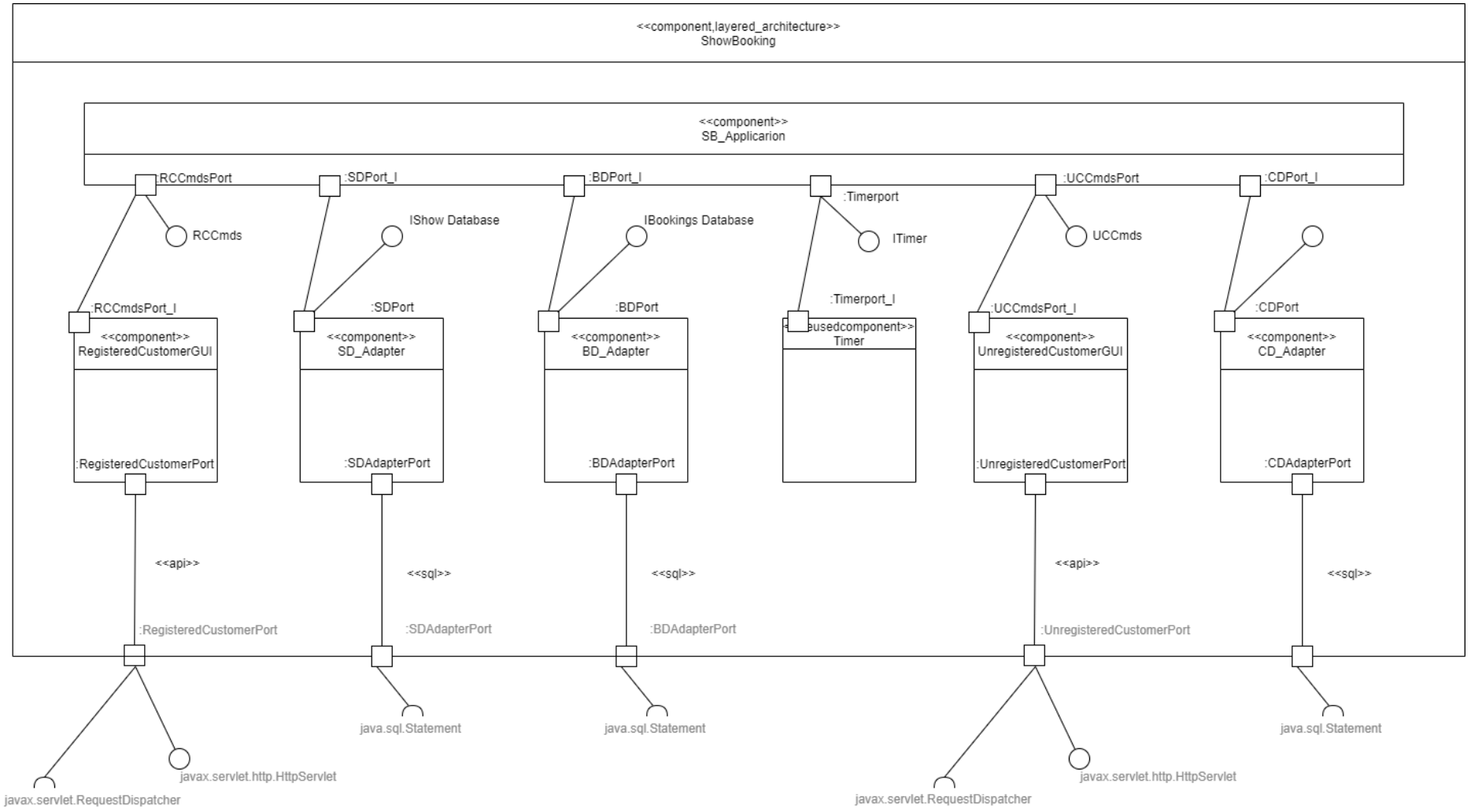


Port types and interface relations for SB_Achive

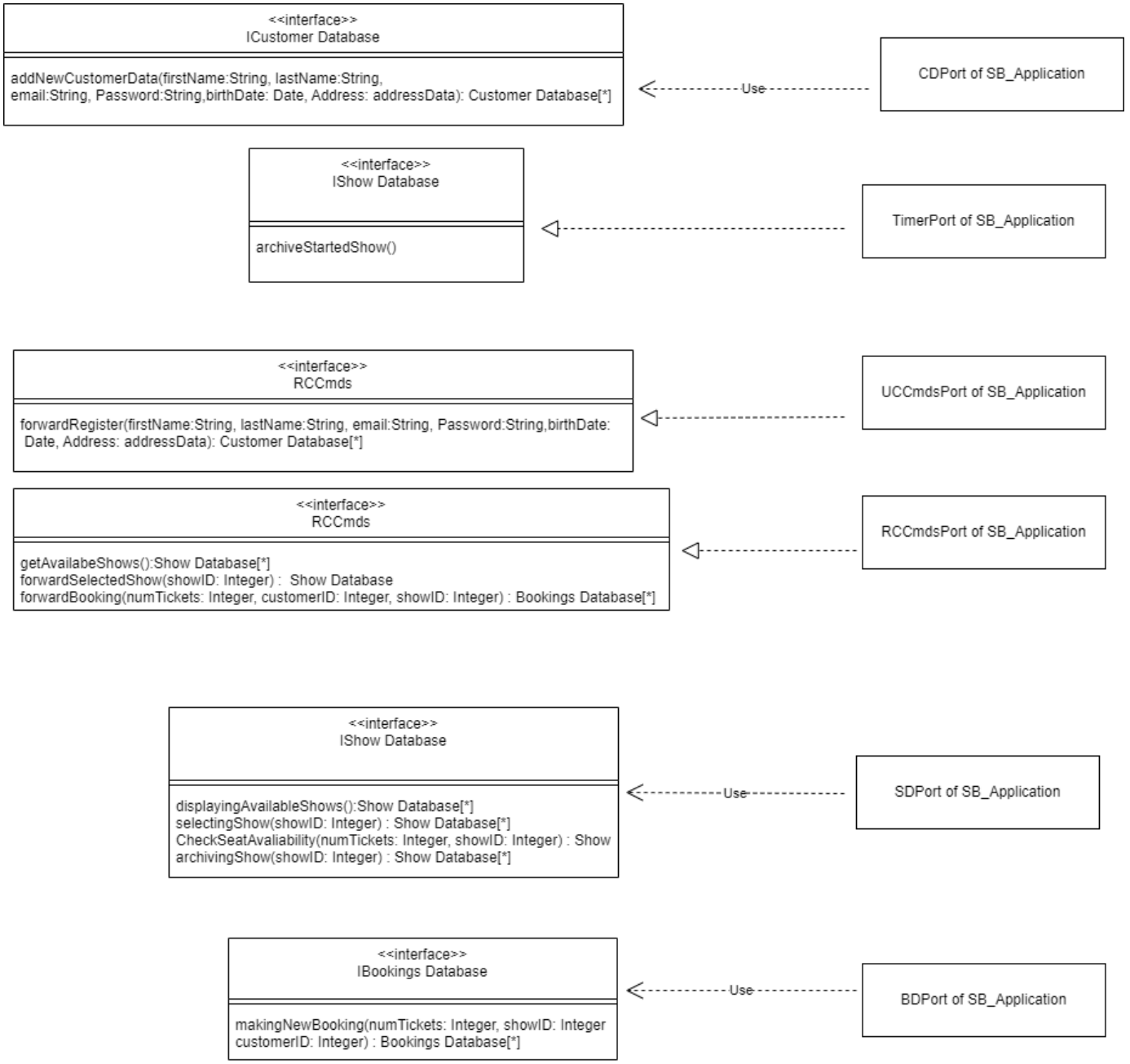




Merged Architecture



Refining App_IF Interface Classes



Refining tech_IF Interface Classes

Considered Interface in subproblem architecture	technical Interface
<<api>> javax.servlet.RequestDispatcher in SB_Register	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.RequestDispatcher in SB_Book and SB_Display	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.http.HttpServlet in SB_Register	<<api>> SB{forward}
<<api>> javax.servlet.http.HttpServlet in SB_Book and SB_Display	<<api>> SB{forward}
<<sql>> java.sql.Statement in SB_Register	<<call_return, sql>> SB!{executeQuery, executeUpdate}
<<sql>> java.sql.Statement in SB_Book	<<call_return, sql>> SB!{executeQuery, executeUpdate}
<<sql>> java.sql.Statement in SB_Display	<<call_return, sql>> SB!{executeQuery, executeUpdate}

Refining adapter_IF Interface Classes

There are no HAL components in the subproblem architechtures. Hence, there are no ‘adapter_if' interface classes that need to be refined.

Customer Database

CustomerID	FirstName	LastName	Email	BirthDate	Address	Password

Show Database

ShowID	Name	Date	Price	AvailableSeats	isArchived

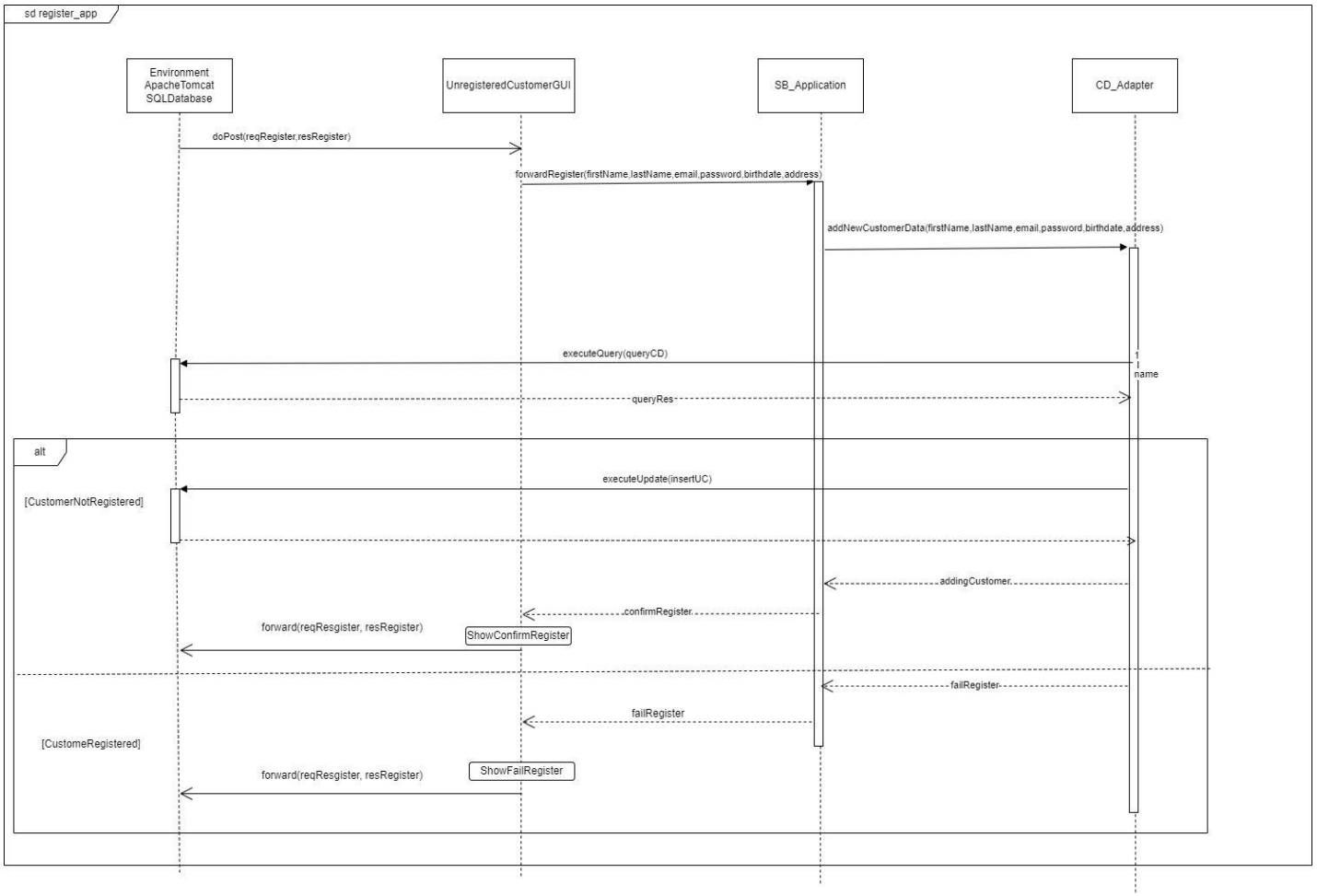
Booking Database

BookingID	CustomerID	NumofTickets	BookingDate	TotalPrice	ShowID

D2

Inter-Component Interaction

Register



query CD

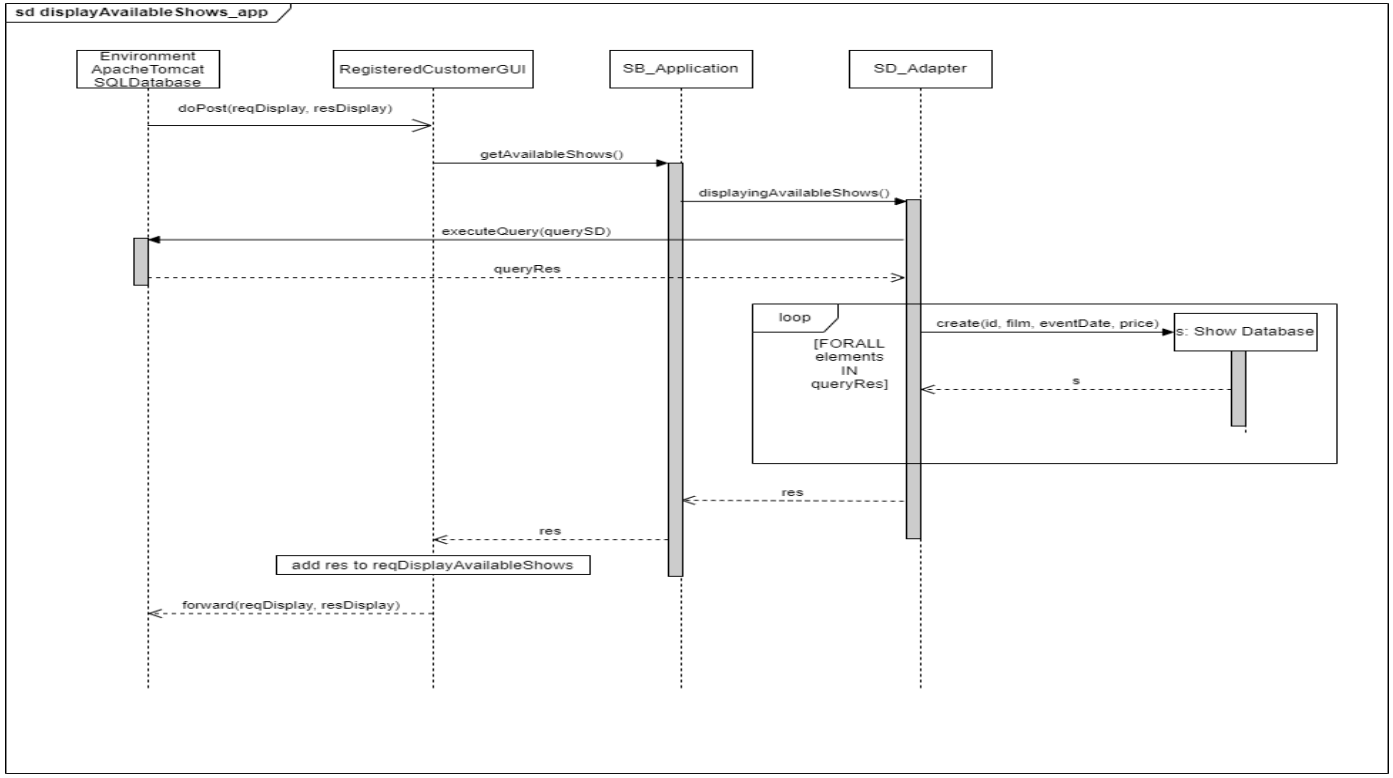
SELECT * FROM CustomerDatabase WHERE (EmailAddress = "email")

insert UC

INSERT INTO CustomerDatabase (FirstName, LastName, EmailAddress, BirthDate, Address, Password)

VALUES ("FirstName", "LastName", "EmailAddress", "BirthDate", "Address", "Password")

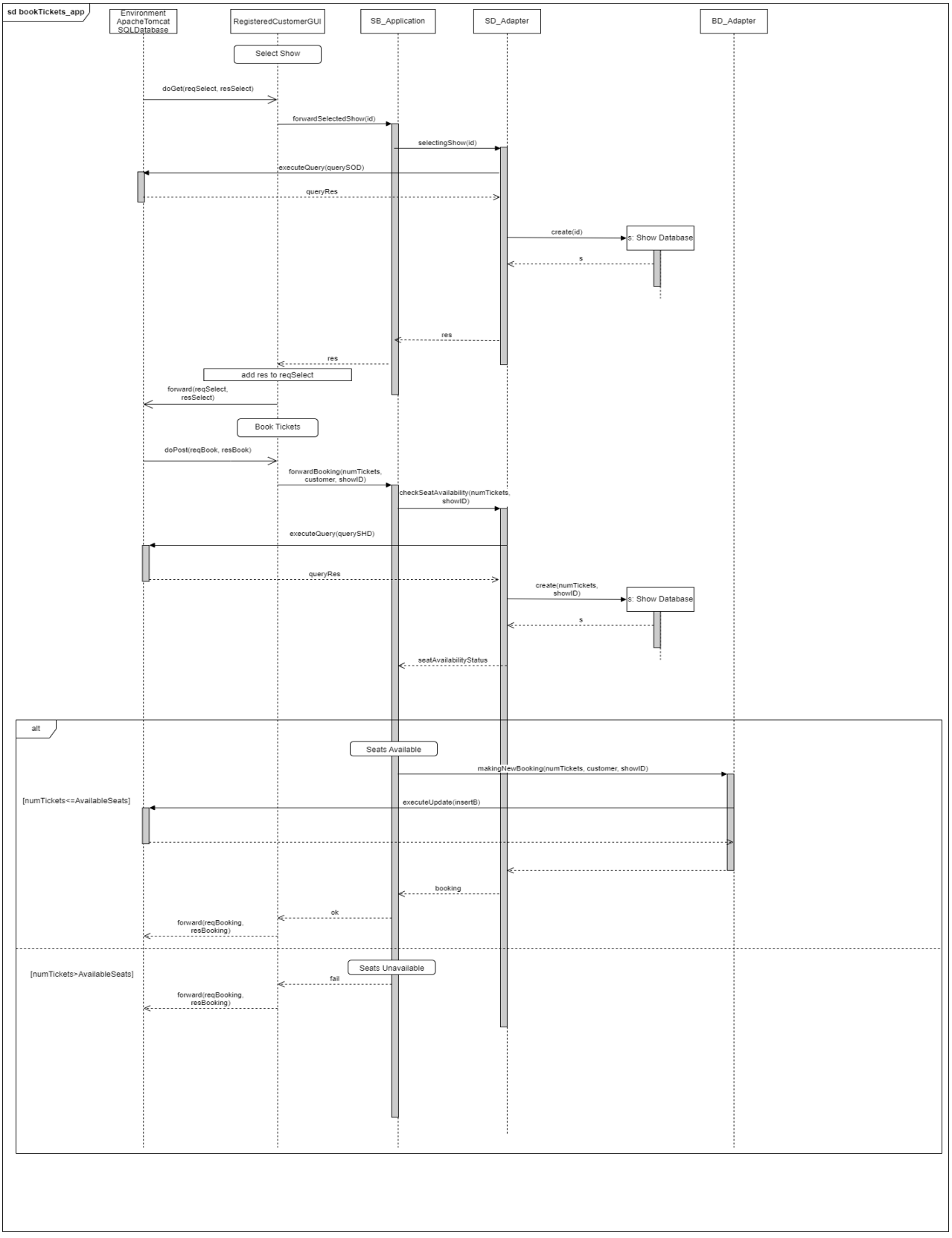
Display



query SD

SELECT * FROM ShowDatabase WHERE (CURRENT_TIMESTAMP<Date) ORDER BY Date

Book


sd bookTickets_app
Environment
ApacheTomcat
SQLDatabase
RegisteredCustomerGUI
SB_Application
SD_Adapter
BD_Adapter
Select Show
doGet(reqSelect, resSelect)
forwardSelectedShow(id)
selectingShow(id)
executeQuery(querySOD)
queryRes
create(id)
s: Show Database
s
res
res
add res to reqSelect
forward(reqSelect,
resSelect)
Book Tickets
doPost(reqBook, resBook)
forwardBooking(numTickets,
customer, showID)
checkSeatAvailability(numTickets,
showID)
executeQuery(querySHD)
queryRes
create(numTickets,
showID)
s: Show Database
s
seatAvailabilityStatus
alt
Seats Available
makingNewBooking(numTickets, customer, showID)
[numTickets<=AvailableSeats]
executeUpdate(insertB)
booking
ok
forward(reqBooking,
resBooking)
[numTickets>AvailableSeats]
Seats Unavailable
fail
forward(reqBooking,
resBooking)

QUERYSOD

SELECT * FROM ShowDatabase WHERE (ID=ShowDatabase.ID)

QUERYSHD

SELECT AvailableSeats FROM ShowDatabase WHERE (ShowID=ShowID AND NumTickets<=AvailableSeats)

QUERYBD

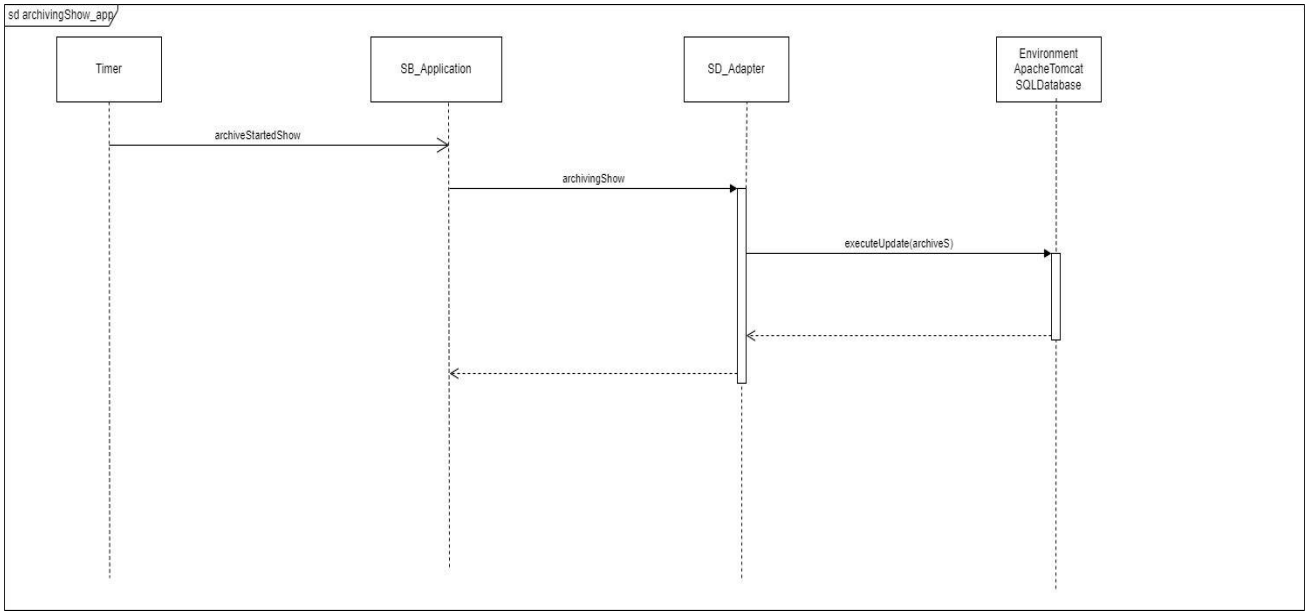
SELECT * FROM BookingDatabase WHERE (BookingDatabase.ShowID=ShowID)

INSERTB

INSERT INTO BookingDatabase (CustomerID, NumofTickets, BookingDate, TotalPrice, ShowID)

VALUES ("Customer, "NumTickets","CURRENT_Date","TotalPrice","ShowID")

Archive

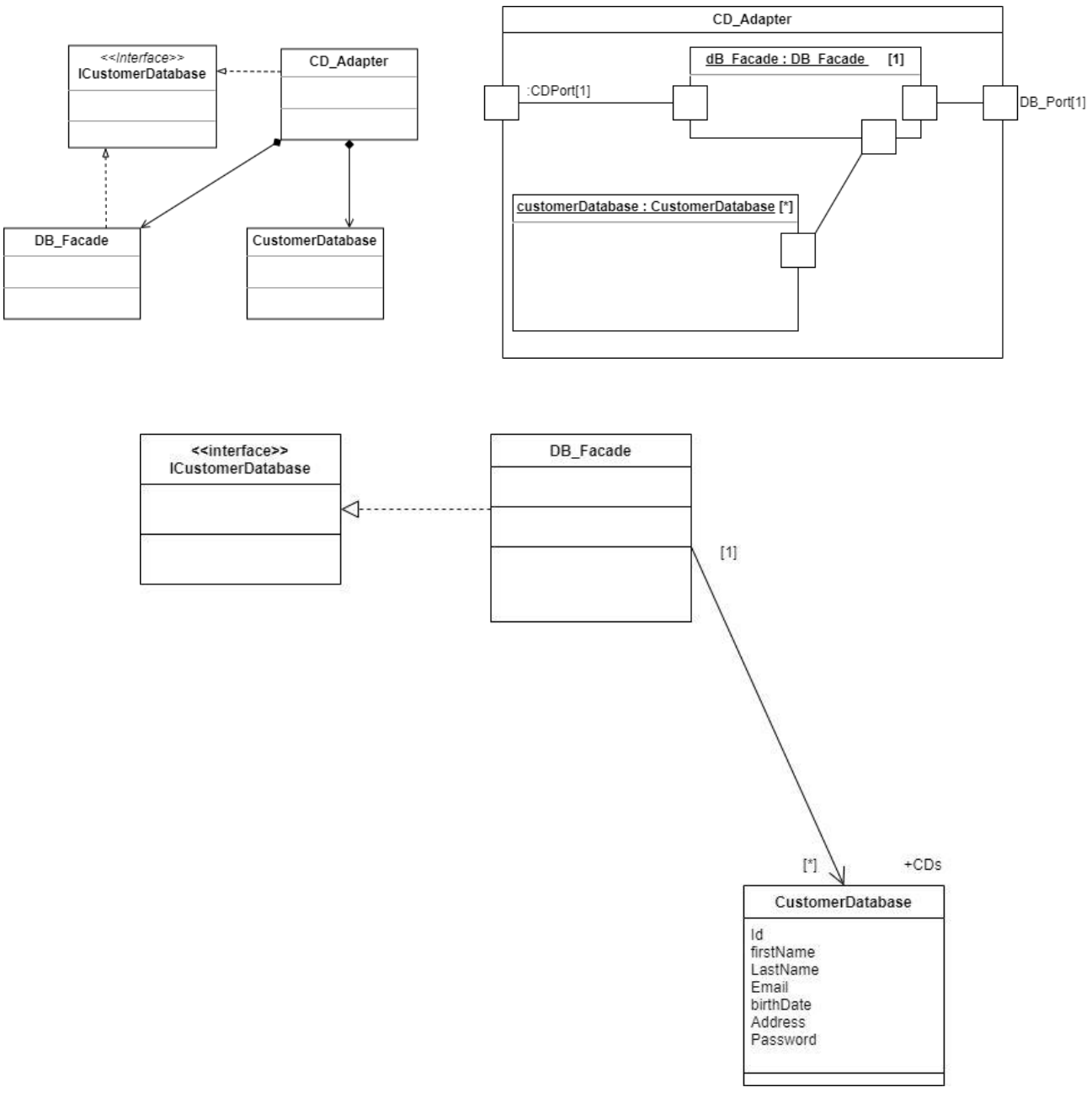


ArchiveS

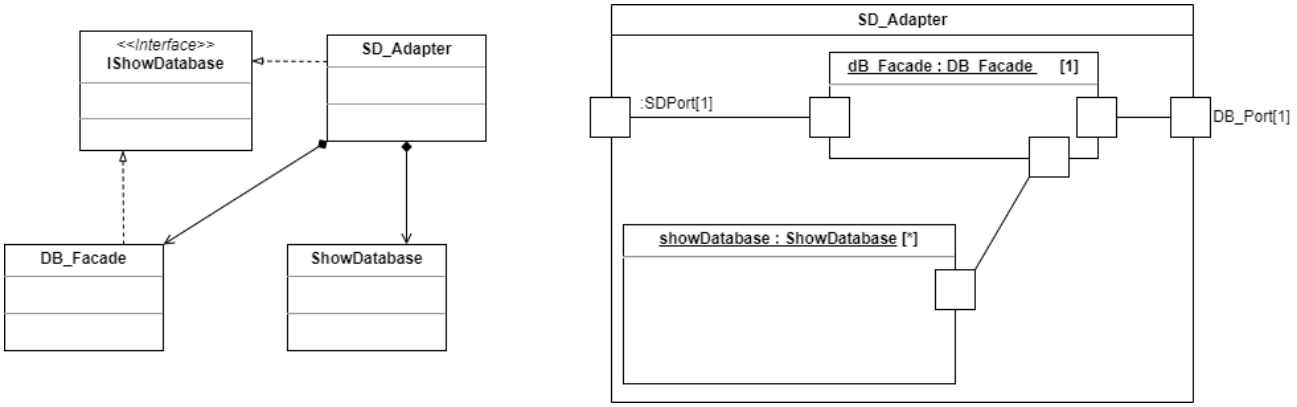
UPDATE ShowDatabase SET isArchived=TRUE WHERE (CURRENT_TIMESTAMP>Date)

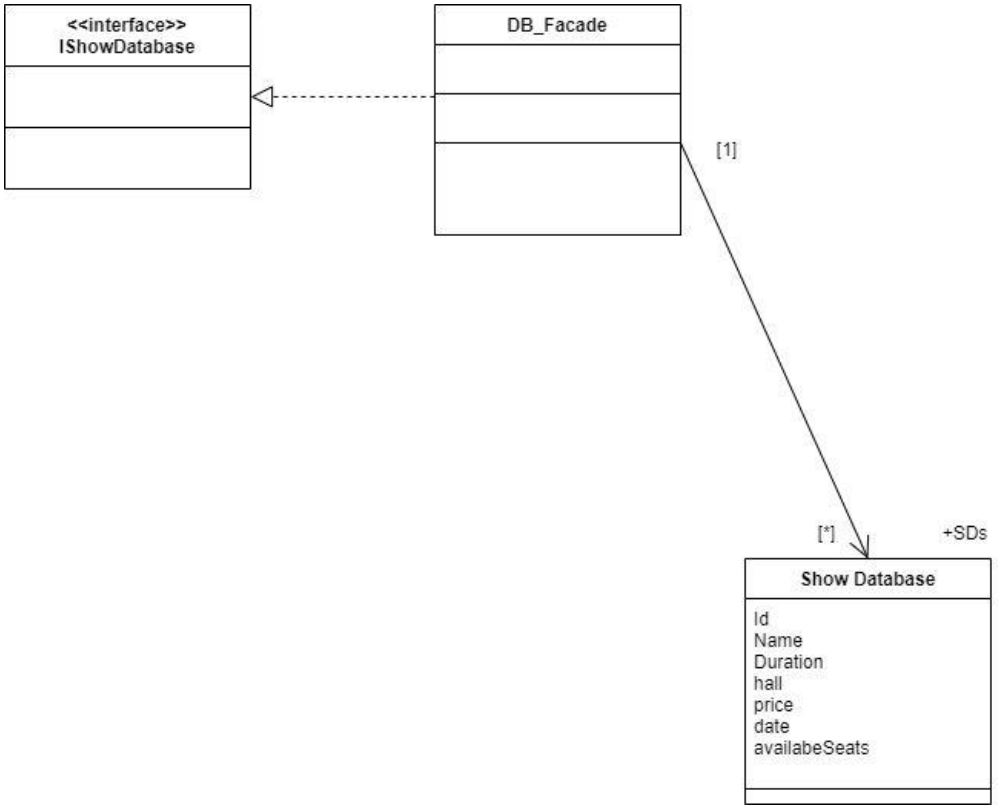
Preliminary Architecture

Preliminary Architectural Description of CD_Adapter

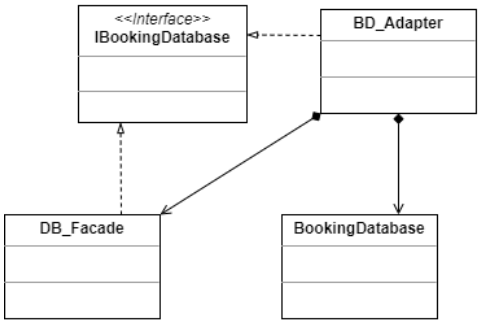


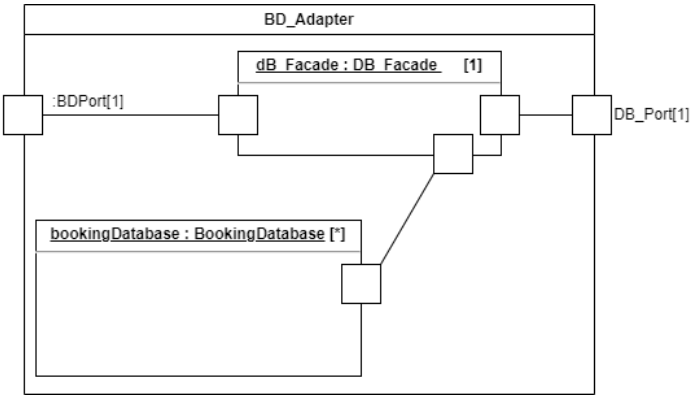
Preliminary Architectural Description of SD_Adapter

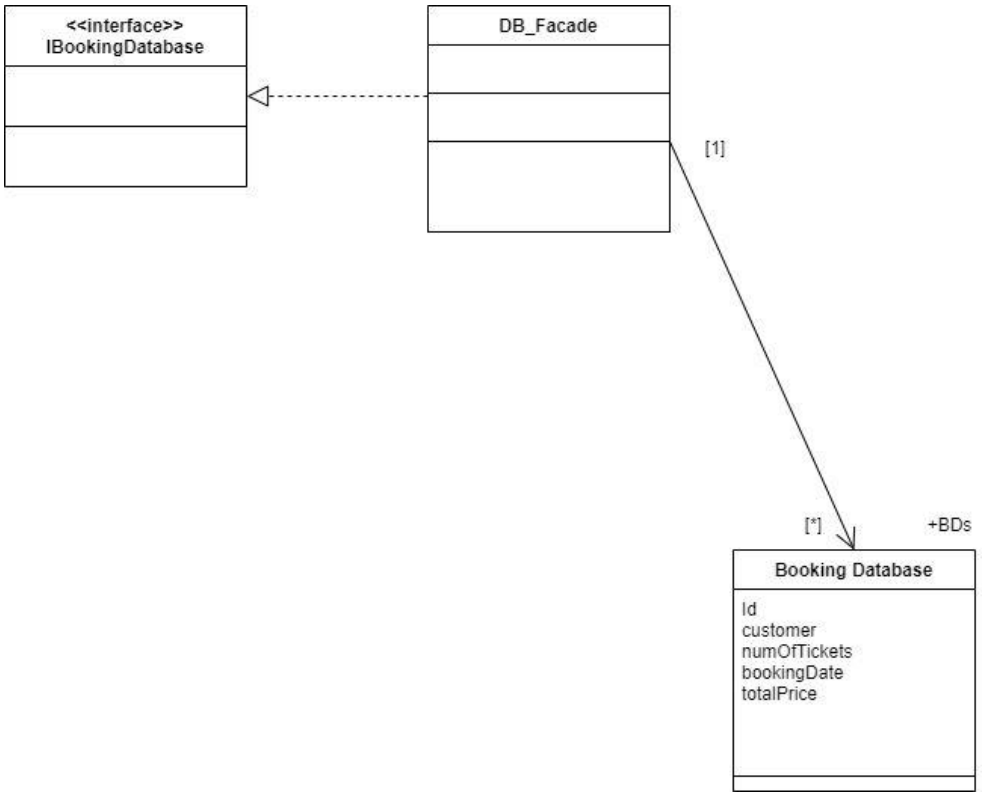




Preliminary Architectural Description of BD_Adapter







D3

sd register_app:

There is no need to specify an intra-component interaction.

The operations of the components are already described in Step D2 and dont need to be decomposed further.

no additional methods, objects or queries need to be further specified

sd displayAvailableShows_app:

There is no need to specify an intra-component interaction.

The operations of the components are already described in Step D2 and dont need to be decomposed further.

no additional methods, objects or queries need to be further specified

sd bookTickets_app:

TThere is no need to specify an intra-component interaction.

The operations of the components are already described in Step D2 and dont need to be decomposed further.

no additional methods, objects or queries need to be further specified

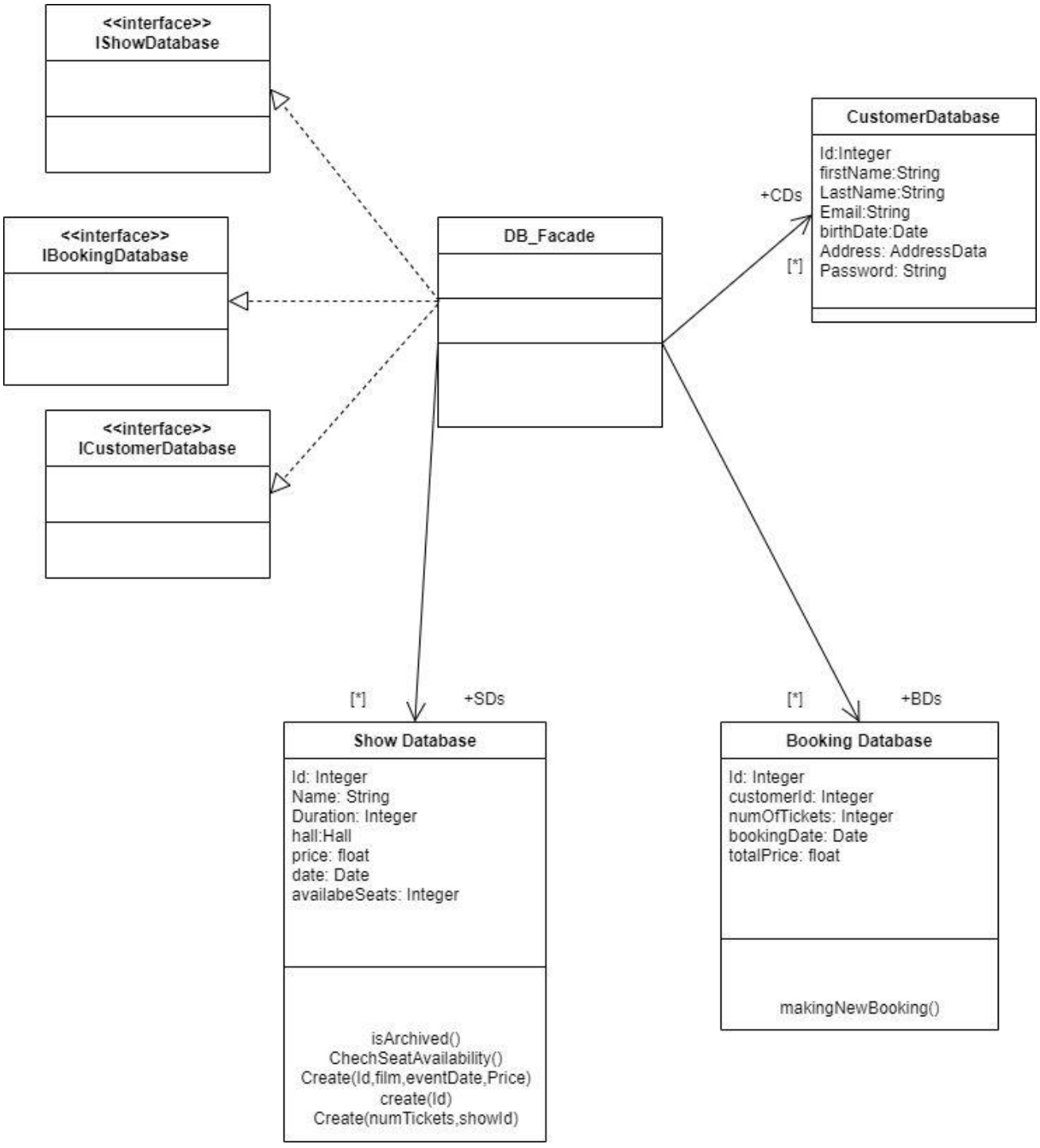
sd archivingShow_app

There is no need to specify an intra-component interaction.

The operations of the components are already described in Step D2 and dont need to be decomposed further.

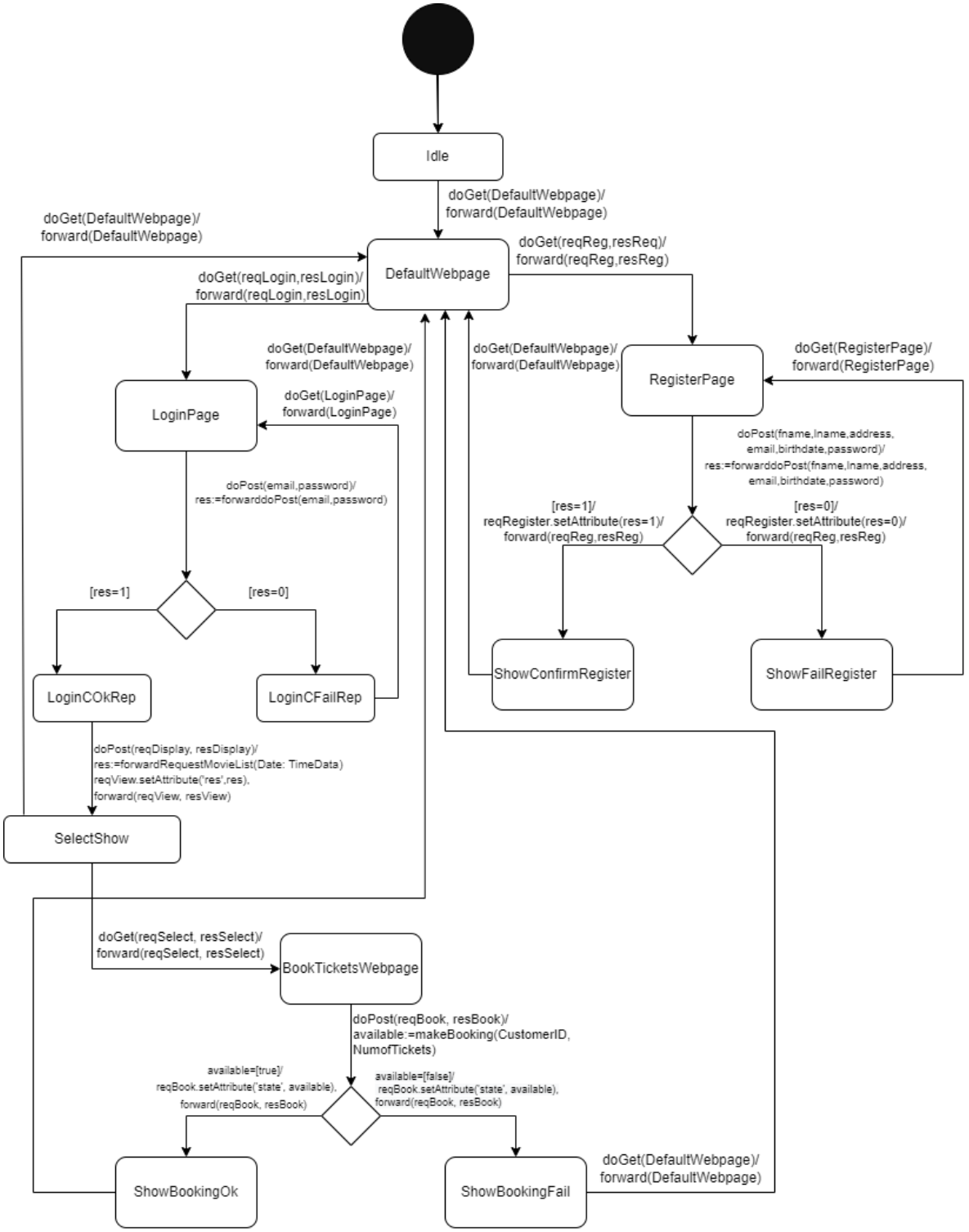
no additional methods, objects or queries need to be further specified

Final Architectural Description



D4

State Machine



Program Notes

1) The Database Schema will be created automatically using JPA and ORM concept but we have also provided all the tables and the queries in our database file
2) in the file you would find a docker-compose file which will can be running to create a my sql database container so it’s required to install docker for that or you can ignore it completely but then change the database properties in the application.properties file to suit your local mysql database
3) All project dependencies and plugins in the pom.xml will be downloaded automatically
4) Our Server works with extra functionalities than the required as the staff member can add shows with an interface (staffmembergui)
5) In case of Testing (Component Test , Unit Test) we use docker test mysql container to create mysql database for testing purpose so for checking all the test cases make sure to install docker locally on your machine and all the test containers will be created in the run time automatically

Glossary

Name	Type	Description	Source
A			
Address	attribute	represents the address of the customer	Class model, state machine GUI
api	technical phenomenon	the way of communicating with the machine	TCD, Class model
ApacheTomcat	connectionDomain	An Open Source JSP and Servlet Container from the Apache Foundation	TCD
availableShows	phenomenon	All available Shows in the Database	CD, pdDisplay, sdDisplay
available	message	Returns true if a seat is available for a Screening and False if no	state machine GUI
addNewCustomerData	phenomenon, auxiliary function	add customers data that was entered by the customer to register, and add it to the database	CD, pdRegister, sdRegister, sdIC_Register
archivingShow	phenomenon, auxiliary function	sent by the machine to the database to archive a show	CD, pdArchive, sdArchive, Class model, sdIC_Archive
archivedShow	phenomenon	feedback from the database to the machine in case of archiving of a show	CD, pdArchive, sdArchive
archiveStartedShow	auxiliary function	represents a function the automatically archives shows 15 minutes after they have started	Class model, sdIC_Archive
availableSeats	attribute	represents available seats in a show	Class model
addCustomer	method	post our form in the database to create a new customer	implementation
addShow	method	adds new show to database	implementation
ArchiveS	message	Java API function to send an SQL update command to a MySQL database.	

B			
birthDate	attribute	represents the birth date of the customer	Class model,state machine GUI
BD_adapter	component	adapter for using the booking database	subArchBook, globalArch, sdIC_Book
BDPort	component	port for Booking database	subArchBook, globalArch
BDPort_I	component	port for booking Database	subArchBook, globalArch
BDAdapterPort	component	port for Booking Database Adapter	subArchBook, globalArch
Booking Database	lexical Domain	Saves the data of the bookings made by customers	CD, sdBook
bookTickets	phenomenon, state predicate	allows the customer to book tickets by specifying the amount of tickets and which seats they want	CD, sdBook, Class model,sdIC_book
back	message	to go back	state machine GUI
BookingSuccess	state predicate	When the booking is a success	sdBook
BookingFail	state predicate	When the booking is a failure	sdBook
bookingDate	attribute	the date the booking was made	Class model
BookWebpage	state	indicates the booking page	state machine GUI
C			
call_return	technical phenomenon	call return	TCD
CD_Adapter	component	adapter for using the customer database	subArchRegister, globalArch, sdIC_Register,
CDPort	component	port for Customer Database	subArchRegister, globalArch
CDPort_I	component	port for Customer Database	subArchRegister, globalArch
CDAdapterPort	component	Port for customer database adapter	subArchRegister, globalArch

customerCredentialsValidation	phenomenon	machine validating the customer's credentials in the database whether they want to register, login or logout	CD
cancelShow	phenomenon	provided Employee functionality to delete a Specific show and all its bookings.	CD
customer	attribute	represents the customer that booked the ticket	Class model
customerID	attribute	represents the customer	state machine GUI
customerCredentialsValiodated	phenomenon	database responding to the machine in case of success in registering the customer or logging in or out of the app or failure in doing so.	CD
createNewShow	phenomenon	employee orders the machine to create a new show	CD
customerData	phenomenon	feedback given by the database to the machine in case of newly added customer data	CD, pdRegister, sdRegister
Customer Database	lexical Domain	Database which shall contain all the customers attributes.	CD, pdRegister, sdRegister, Class model
creatingNewShow	phenomenon	Functionality which allow the machine to save new show in the show database	CD
createdNewShow	phenomenon	Database response in case of creating new show successfully or failure in creating it	CD
CustomerRegistered	state predicate, auxiliary function	The customer is already registered in our database and so cannot register once more	sdRegister, sdIC_Register
CustomerNotRegistered	state predicate, auxiliary function	The customer is not registered and so will now be registered in our database as a new customer	sdRegister, sdIC_Register
CustomerWebBrowser	connectionDomain	Web browser used by customer	TCD
CheckSeatAvailability	auxiliary function	function to check the available seats in a show	class model, sdIC_Book

createCustomer	method	return the form which needs to be filled by user	implementation
D			
DefaultWebpage	state	Indicates the starting page	state machine GUI
deleteBookingsforShow	phenomenon	Functionality which allows the machine to delete all the bookings on a specific show .	CD
deletedBookingsforSho w	phenomenon	Database response in case of deletion of all bookings on specific shows.	CD
date	attribute	represents the date	Class model, state machine GUI
delete1YearOldShow	phenomenon	after a show reaches a year in the database, the machine deletes it automatically by sending this order to the database	CD
deletingShow	phenomenon	the machine requests to delete a specific show from the database.	CD
deletedShow	phenomenon	Database response in case of deletion specific shows.	CD
displayAvailableShows	phenomenon	Customer request functionality to return all the available offers	CD, pdDisplay, sdDisplay, Class model
displayingAvailableSho ws	phenomenon, auxiliary function	Machine request to the database to return all the available offer from it	CD, pdDisplay, sdDisplay, sdIC_Display
doGet	technical phenomenon	the way of displaying the data to the webpageCustomer from the database through the machine	TCD, state machine GUI
doPost	technical phenomenon, auxiliary function	the way getting data from the webpageCustomer	TCD, sdIC_Register, sdIC_Display, sdIC_Book,state machine GUI
E			
eventDate	attribute	Represents the date of the show	Class model

executeQuery	technical phenomenon, message	executes a required query in the database using SQL	TCD, sdIC_Register, sdIC_Book, sdIC_Display
executeUpdate	technical phenomenon, message	executes a required update in the database using SQL	TCD, sdIC_Register, sdIC_Book, sdIC_Archive
Email	attribute	represents the email of the customer	Class mode, state machine GUIl
eMailNotification	phenomenon	the machine sends a message automatically to the customer that a show that they booked a ticket for has been canceled.	CD
Employee	biddable Domain	person who is working for the cinema office	CD
F			
film	attribute	Represents the details of the film of the show	Class model
firstName	attribute	Represents the first name of the customer	Class model
feedbackRC	phenomenon	capture all the feedback going back to a registered customer	CD
feedbackUC	phenomenon	capture all the feedback going back to an unregistered customer	CD
feedbackE	phenomenon	capture all the feedback going back to the Employee	CD
forward	technical phenomenon	forwards a message	TCD,state machine GUI
forwardRegister	phenomenon, auxiliary function	WebpageCustomer sends to the machine the customer's request to register	pdRegister, sdRegister, Class model, sdIC_Register
forwardSelectedShow	phenomenon, auxiliary function	webpage customer forwards to the machine the order to select a show by the customer	sdBook, Class model, sdIC_Book
forwardBooking	phenomenon, auxiliary function	webpageCustomer forwards to the machine the order to make a booking by the customer	sdBook, Class model, sdIC_Book

failRepresentation	phenomenon	Webpage customer displays to the customer that booking the ticket(s) was a failure	sdBook
G			
getAvailableShows	phenomenon, auxiliary function	the WebpageCustomer forwarding the customer's request to the machine to display the available shows	pdDisplay, sdDisplay, Class model, sdIC_Display
gui	technical phenomenon	graphical user interface, what the customer sees when he uses the website	TCD
getMain()	method	return to main page	implementation
getShows()	method	allows user to display a list of the shows	implementation
getShow()	method	return a list of shows from database	implementation
getSavedShowFromDatabase()	method	asserts that all parameters are saved correctly in the database	testing
getNewCustomer()	method	assure that values are saved correctly in our database	testing
H			
http	technical phenomenon	Hypertext Transfer Protocol, is an application-layer protocol, for by which the customerWebBrowser communicates with ApacheTomcat	TCD
hall	attribute	represents the hall the show is held in	Class model
I			
Id	attribute	Represents the unique ID of the customer	Class model
		Represents the unique ID of the show	
		Represents the unique ID of the booking	
Idle	state	indicates that the server waits for incoming request	state machine, GUI

isArchived	auxiliary function	function to check whether show is archived or not	Class model
IShow Database	interface	used to trigger the internal operation	subArchDisplay, subArchBook, subArchArchive, globalArch
ITimer Database	interface	used to trigger the internal operation "archiveStartedShow" periodically	subArchArchive, globalArch
IBookings Database	interface	used to trigger the internal operation	subArchBook, globalArch
ICustomer Database	interface	used to trigger the internal operation	subArchRegister, globalArch
J			
K			
L			
LastName	attribute	Represents the last name of the customer	Class model
lname	attribute	Represents the last name of the customer	state machine GUI
LoginCOkRep	state	indicates if the login was successful	state machine GUI
LoginCFailRep	state	indicates if the login failed	state machine GUI
login	phenomenon	customer orders the machine to login	CD
loginC			
logout	phenomenon	customer orders the machine to logout	CD
loginpage	state	Indicates the login page	state machine GUI
LCUnregistered Customer	life-cycle	Life-cycle for one unregistered customer	LC

$LC_{Registered\ Customer}$	life-cycle	Life-cycle for one registered customer	LC
$LC_{Show\ Booking}$	life-cycle	Combined life-cycle(all customers and the internal operation)	LC
M			
makingNewBooking	phenomenon, auxiliary function	machine sends the booking that the customer wants to make to the database responsible for saving the booking	CD, sdBook, Class Model, sdIC_Book
madeNewBooking	phenomenon	database responds to the machine with the status of the booking made.	CD, sdBook
N			
numOfTickets	attribute	represents number of Tickets in a booking in the database	Class model,state machine GUI
O			
okRepresentation	phenomenon	Webpage customer displays to the customer that booking the ticket(s) was a success	sdBook
P			
Password	attribute	represents the password of the customer	Class model, state machine GUI
price	attribute	Represents the price of the show	Class model
Q			
querySHD	message	Java API function to execute an SQL query command to a MySQL database	sdIC_Book
querySOD	message	Java API function to execute an SQL query command to a MySQL database	sdIC_Book
querySD	message	Java API function to execute an SQL query command to a MySQL database	sdIC_Display

queryRes	message	Java API function to execute an SQL query command to a MySQL database	sdIC_Display, sdIC_Register, sdIC_Book
queryBD	message	Java API function to execute an SQL query command to a MySQL database	sdIC_Book
queryCD	message	Java API function to execute an SQL query command to a MySQL database	sdIC_Register
R			
register	phenomenon	customer request to register itself in our website	CD, pdRegister, sdRegister, Class model
RegisteredCustomerGUI	component	web interface for registered guests	subArchDisplay, subArchBook, globalArch, sdIC_Book, sdIC_Display
RCCmds	interface	Registered Customer Commands	subArchDisplay, subArchBook, globalArch
RCCmdsPort	component	port for registered customer commands	subArchDisplay, subArchBook, globalArch
RCCmdsPort_I	component	port for registered customer commands	subArchDisplay, subArchBook, globalArch
RegisteredCustomerPort	component	port for registered customer	subArchDisplay, subArchBook, globalArch
RegisterPage	state	Indicates the register page	state machine GUI
Registered Customer	biddable Domain	the targeted customer who will use our system and part from our environment	CD, pdDisplay, sdBook, sdDisplay, TCD
registerOK	phenomenon	Machine sends to WebpageCustomer the success of the registration process	pdRegister, sdRegister, Class model
registerOKRep	phenomenon	WebpageCustomer forwards the machine's confirmation of	pdRegister, sdRegister

		success of registration by displaying it to the customer	
registerCOKRep	state	WebpageCustomer forwards the machine's confirmation of success of registration by displaying it to the customer	state machine GUI
registerFail	phenomenon	Machine sends to WebpageCustomer the failure of the registration process	pdRegister, sdRegister, Class model
registerFailRep	phenomenon	WebpageCustomer forwards the machine's failure of registration message by displaying it to the customer	pdRegister, sdRegister
registerCFailRep	state	WebpageCustomer forwards the machine's failure of registration message by displaying it to the customer	state machine GUI
reqLogin	message	Required user input	state machine GUI
reqReg	message	Required user input	state machine GUI
reqSelect	message	Required user input	state machine GUI
reqDisplay	message	Required user input	state machine GUI
reqBook	message	Required user input	state machine GUI
resLogin	message	Response to an user input	state machine GUI
resReg	message	Response to an user input	state machine GUI
resSelect	message	Response to an user input	state machine GUI
resDisplay	message	Response to an user input	state machine GUI
resBook	message	Response to an user input	state machine GUI
returnStaffGUI	method	return a form where the staff member adds new shows	implementation
registerMySQLPropertie s	method	connect our app with the database	testing
s			
Seats Available	state predicate	displays if the selected seats to be booked are available	sdIC_Book

Seats Unavailable	state predicate	displays if the selected seats to be booked are not available	sdIC_Book
Show Booking	machine	Our main machine that helps the customer register, book tickets, display shows…….etc.	CD,TCD
SB_Application	machine, component	The show Booking application	subArchDisplay, subArchBook, globalArch, sdIC_Register, sdIC_Display, sdIC_Book, sdIC_Archive
SB_Register	machine	part of the machine that is responsible for registering a customer	pdRegister, sdRegister, Class model
SB_Display	machine	part of the machine that helps the customer display the future shows	pdDisplay, sdDisplay, Class model, subArchDisplay
SB_Book	machine	part of the machine that helps the customer select the show and book tickets for it.	pdBook, sdBook, Class model
SB_Archive	machine	part of the machine responsible for automatically archiving a show after it has started	pdArchive, sdArchive, class model
SD_Adapter	component	adapter for using show database	subArchDisplay, subArchBook, subArchArchive, globalArch, sdIC_Display, sdIC_Book, sdIC_Archive
SDPort_I	component	port for show database	subArchDisplay, subArchBook, globalArch
SDPort	component	port for show database	subArchDisplay, subArchBook, globalArch
SDAdapterPort	component	port for show database adapter	subArchDisplay, subArchBook, globalArch

Show Database	lexical Domain	The Database which stores all our available shows	CD, pdDisplay, sdBook, sdDisplay, sdArchive, Class model
selectShow	phenomenon, state predicate	customer selects the show from the list displayed to him by the machine	CD, pdBook, sdBook, Class model, sdIC_Book
selectingShow	phenomenon, auxiliary function	machine selects the show selected by the customer from the database	CD, pdbook, sdBook, sdIC_Book
selectedShow	phenomenon	database returns the selected show to the machine	CD, pdBook, sdBook
SelectShowDatabase	state	customer selects the show from the list displayed to him by the machine	state machin GUI
showAvailableShows	phenomenon	machines response to the order it was given by the WebpageCustomer to display the available shows	pdDisplay, sdDisplay, Class model
showsRepresentation	phenomenon	WebpageCustomer displays the available shows sent by the machine to the customer	pdDisplay, sdDisplay
ShowBookingOk	state	machine sends to webpageCustomer if it was a success to book the wanted tickets	state machine GUI
ShowBookingFail	state	machine sends to webpageCustomer if it was a fail to book the wanted tickets	state machine GUI
showOk	phenomenon	machine sends to webpageCustomer if it was a success to book the wanted tickets	pdBook, sdBook, Class model
showFail	phenomenon	machine sends to webpageCustomer if it was a failure to book the wanted tickets	sdBook, Class model
ShowDetails	phenomenon	machine send to webpage customer all the available show from the database	sdBook

ShowDetailsRep	phenomenon	WebpageCustomer displays the available shows to customer	sdBook
ShowConfirmRegister	state predicate	Positive feedback for registration	sdIC_Register
ShowFailRegister	state predicate	Negative feedback for registration	sdIC_Register
sql	technical phenomenon	Structured Query Language, used to add to or extract something from the database	TCD
SQLDatabase	causalDomain	Our Database	TCD
staffmembergui	method	let staff member create new show and add specific film ID to it	Implementation
show	object	An instance of the class 'Show Database'	Class model OCL
saveNewcustomer	method	saves the new customer data in the database	implementation
setUp	method	create a trial object which will be saved in the database and will be used in our test	testing
T			
totalPrice	attribute	total price for the booking	Class model
Timer	reused component	given component initiating the internal operation "archiveStartedShow"	subArchArchive, globalArch, sdIC_Archive
TimerPort	component	port for timer	subArchArchive globalArch
TimerPort_I	component	port for timer	subArchArchive, globalArch
testGUI	method	used to check input fields and title after system test	testing
U			
Unregistered Customer	biddable Domain	the targeted person who will use our system and part from our environment	CD, pdRegister, sdRegister,TCD

UnregisteredCustomerGUI	component	web interface for unregistered guests	subArchRegister, globalArch, sdIC_Register
UCCmds	interface	Unregistered Customer Commands	subArchRegister, globalArch
UCCmdsPort	component	port for unregistered Customer	subArchDisplay, subArchBook, globalArch
UCCmdsPort_I	component	port for unregistered Customer	subArchRegister, globalArch
UnregisteredCustomerPort	component	port for unregistered customer adapter	subArchRegister, globalArch
V			
W			
WebpageCustomer	connectionDomain	The connection between a customer and the machine, forwards the customer's orders to the machine and forwards the machine's feedback to the customer.	pdRegister, pdDisplay, pdBook, sdRegister, sdDisplay, sdBook, Class model
webpageCustomer	object	object of the class WebpageCustomer	Class model
X			
Y			
Z			