Online Ticket-Booking App For A Cinema Hall

Software Engineering Lab WS22/23

Nazmul Huda Anik (3115997), MD Anamul Haque (3085596), MD Nazrul Islam Nayem (3099491), Md Shakhawat Hossain (3099378), Md Zafor Iqbal (3108092), Shilan Ali (3090824)

Lab Time: Dienstag, 16:00 - 18:00
Supervisor: Roman Wirtz
Date of submission: 22:01:2023

A1.1 Requirements & Domain Knowledge

Requirements

- **R1:** Customers must be registered with an email address and a password to book a cinema ticket.
- **R2:** To be clearly identifiable an e-mail address can only be used exactly by one customer.
- **R3:** Providing the e-mail address and password customer should be able to log into their account.
- **R4:** Customers should be able to Unsubscribe.
- **R5:** The list of future performances should be available to display to the customers which will be sorted in ascending order of the time of the performances (next first).
- **R6:** Each performance should be clearly identifiable by an ID.
- **R7:** The employees of the cinema should be able to create new performances that will take place in the future by providing the necessary data.
- **R8:** Registered customers will not only be able to book tickets for a show from the list but also be able to specify how many and which seats they want.
- **R9:** Booking a ticket is only possible till 15 minutes before the start of the performance.
- **R10:** A performance should be marked as "archived" after it has already started just to make it still available to the staff but not in the display list.
- **R11:** Employees should be able to cancel any performance, for which all the bookings will also be deleted from the system and the customers will get notified about the cancellation through the provided e-mail address.
- **R12:** Old performances that are more than 1 year old should be deleted along with all the associated bookings for it.

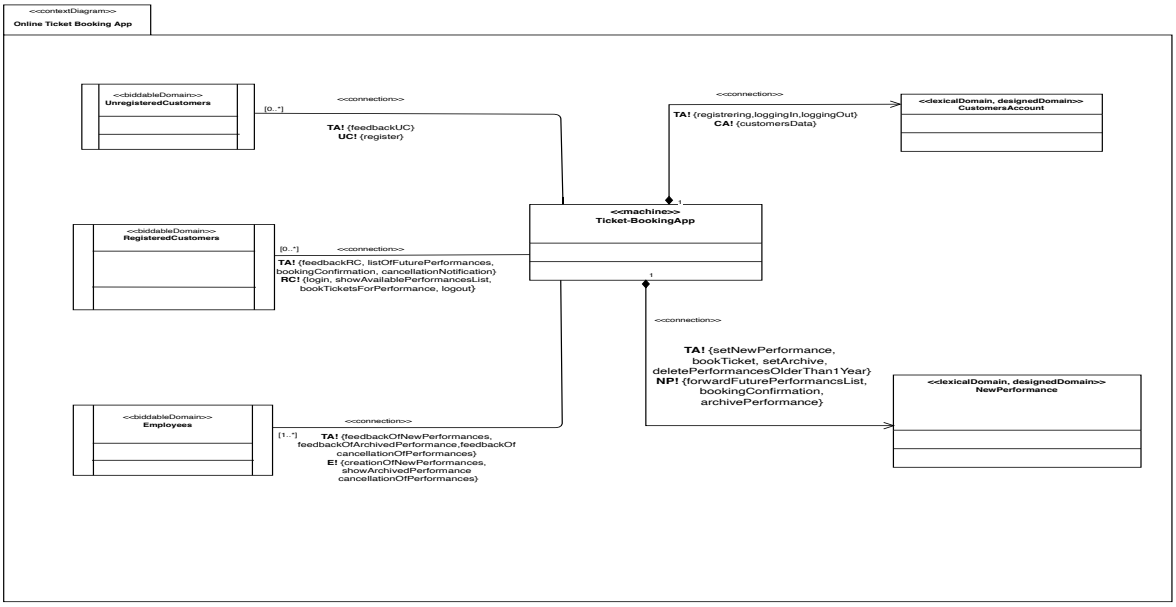
Assumptions

- **A1:** All customer are able to use a web application.
- **A2:** Only self-accessible e-mails will be provided by the customers.
- **A3:** Customers will check this page regularly to find out new ideas.

Facts

- **F1:** A performance consists of the title of the film, the duration, an assigned hall and a date with 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.

A1.2 Context Diagram



A2 Problem Diagram

New R1:

R1: Customers must be registered with an email address and a password to book a cinema ticket.

R2: To be clearly identifiable an e-mail address can only be used exactly by one customer.

New R2:

R5: The list of future performances should be available to display to the customers which will be sorted in ascending order of the time of the performances (next first).

R6: Each performance should be clearly identifiable by an ID.

New R3:

R8: Registered customers will not only be able to book tickets for a show from the list but also be able to specify how many and which seats they want.

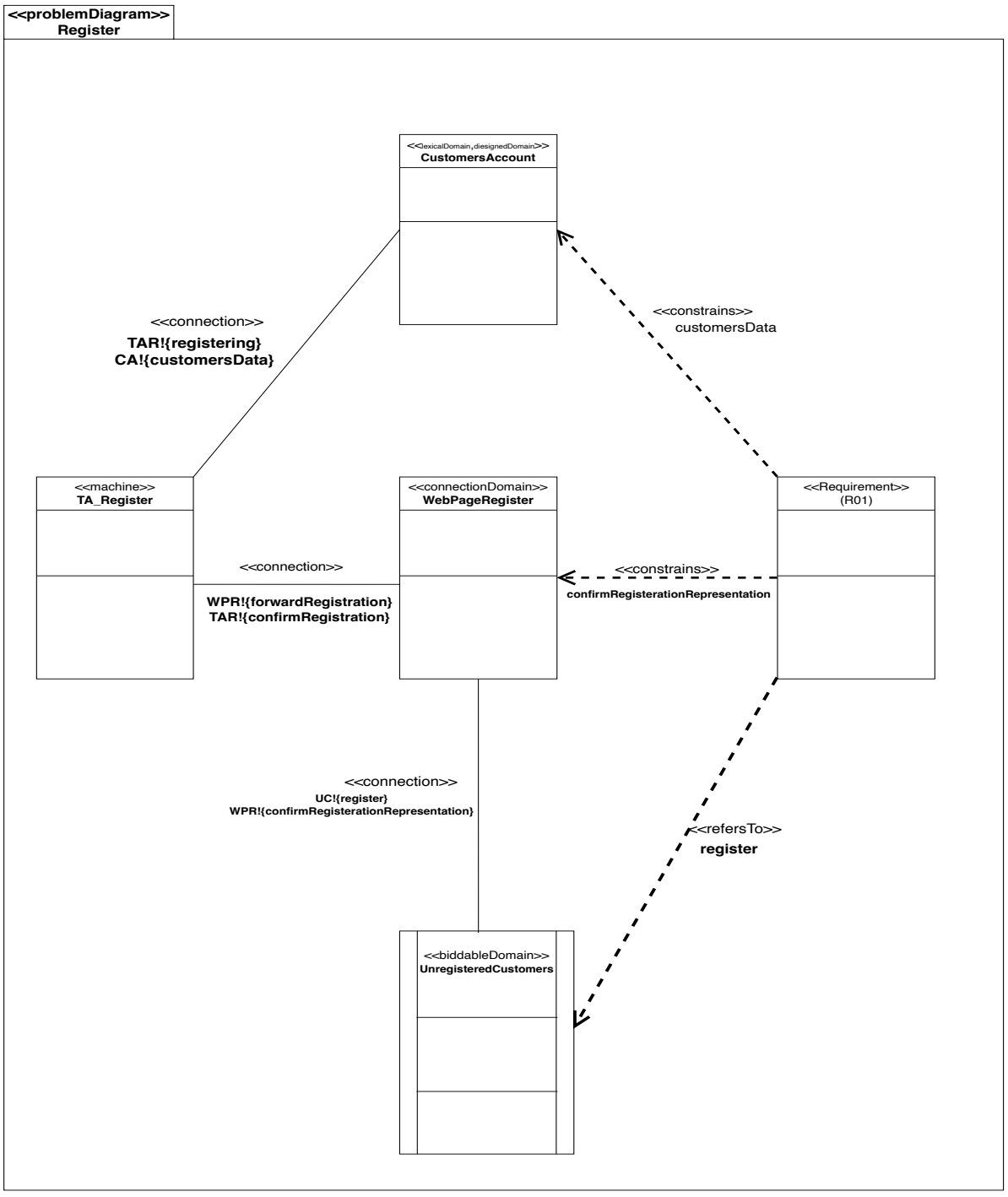
R9: Booking a ticket is only possible till 15 minutes before the start of the performance.

New R4:

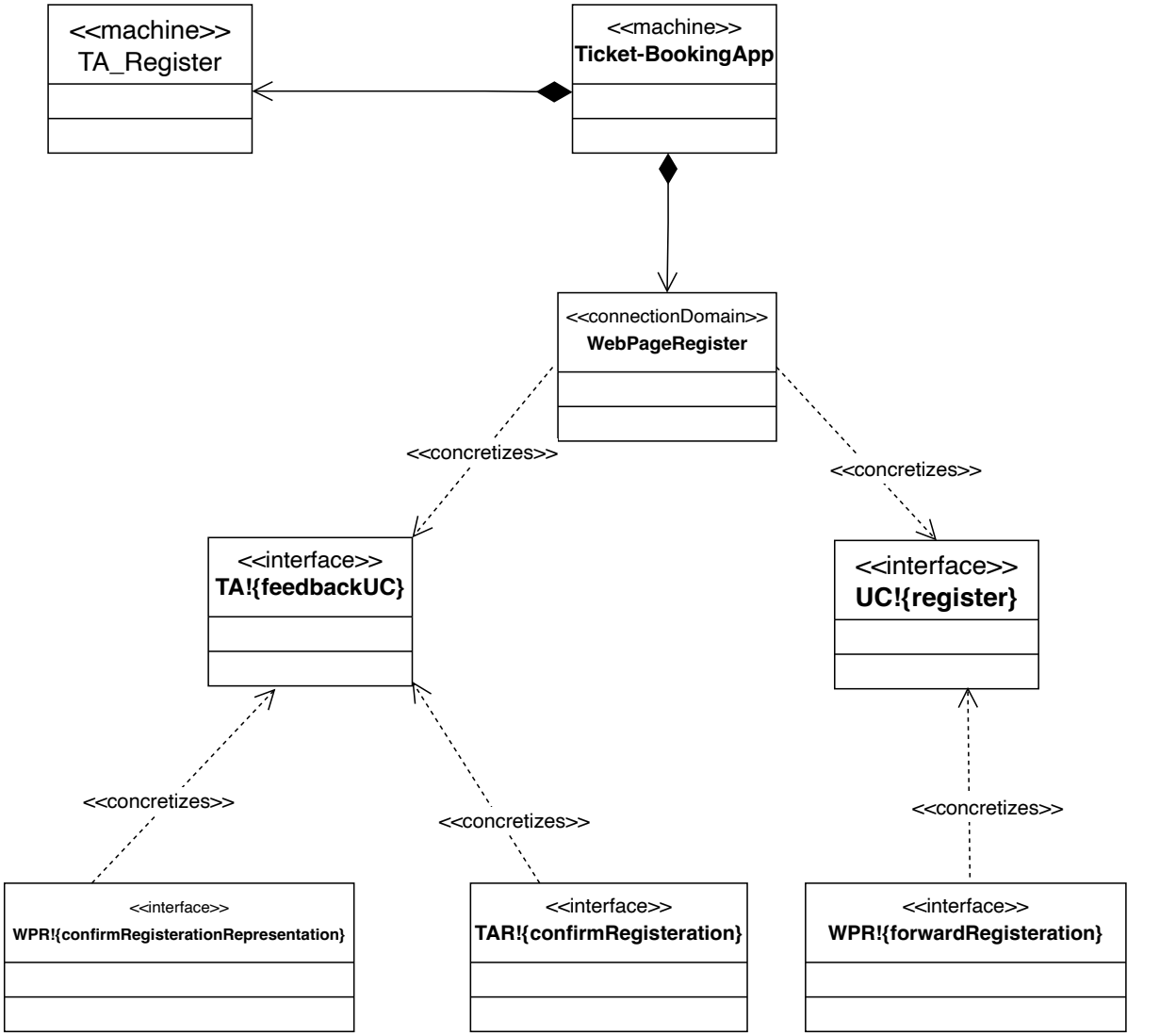
R10: A performance should be marked as "archived" after it has already started just to make it still available to the staff but not in the display list.

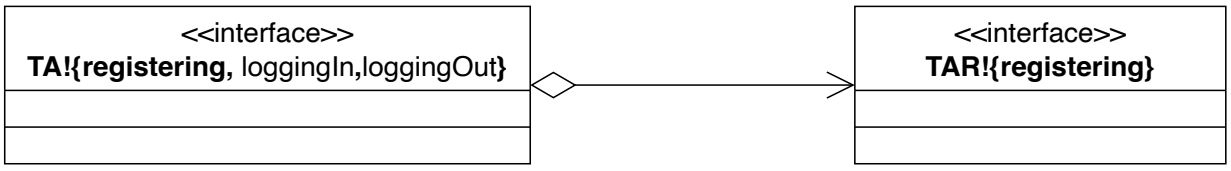
R1 Problem Diagram

Instantiation of Problem Frame "Update2"


<<problemDiagram>>
Register
<<lexicalDomain, designedDomain>>
CustomersAccount
<<connection>>
TAR!{registering}
CA!{customersData}
<<constrains>>
customersData
<<machine>>
TA_Register
<<connectionDomain>>
WebPageRegister
<<Requirement>>
(R01)
<<connection>>
WPR!{forwardRegistration}
TAR!{confirmRegistration}
<<constrains>>
confirmRegisterationRepresentation
<<connection>>
UC!{register}
WPR!{confirmRegisterationRepresentation}
<<refersTo>>
register
<<biddableDomain>>
UnregisteredCustomers

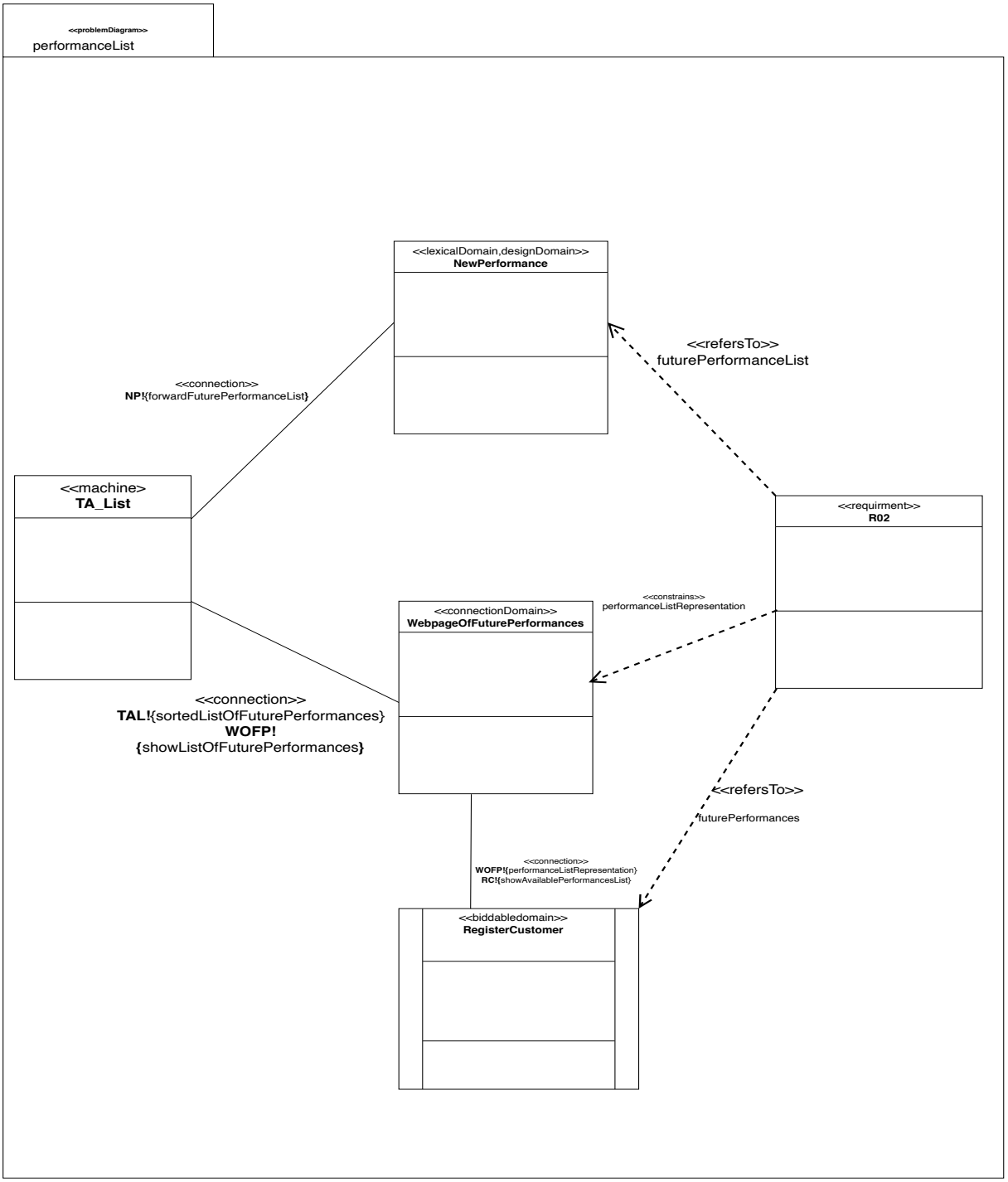
Mapping:R1


<<machine>>
TA_Register
<<machine>>
Ticket-BookingApp
<<connectionDomain>>
WebPageRegister
<<concretizes>>
<<concretizes>>
<<interface>>
TA!{feedbackUC}
<<interface>>
UC!{register}
<<concretizes>>
<<concretizes>>
<<concretizes>>
<<interface>>
WPR!{confirmRegisterationRepresentation}
<<interface>>
TAR!{confirmRegisteration}
<<interface>>
WPR!{forwardRegisteration}

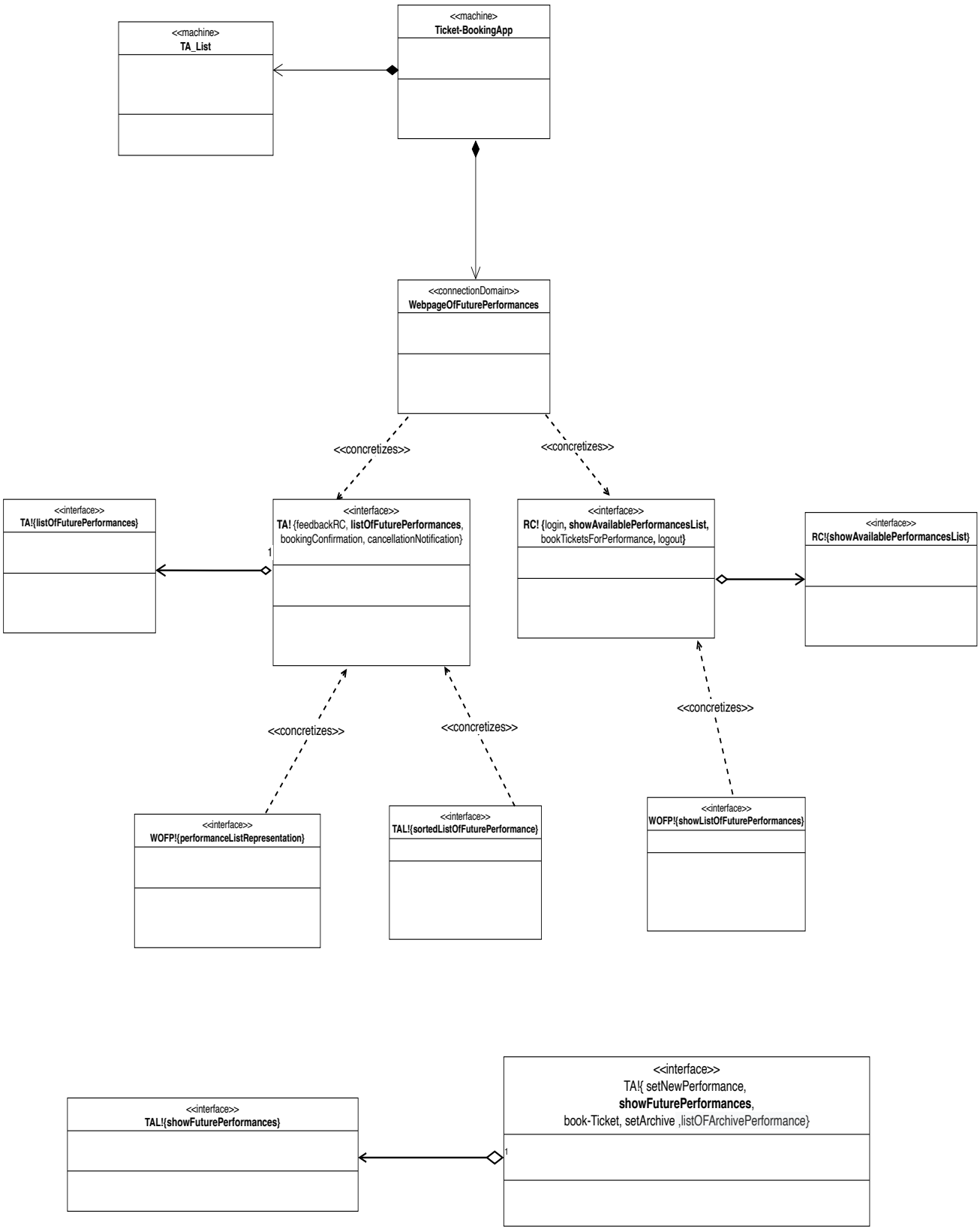

<<interface>>
TA!{registering, loggingIn,loggingOut}
<<interface>>
TAR!{registering}

R2 Problem Diagram

Instantiation of Problem Frame “Query2”

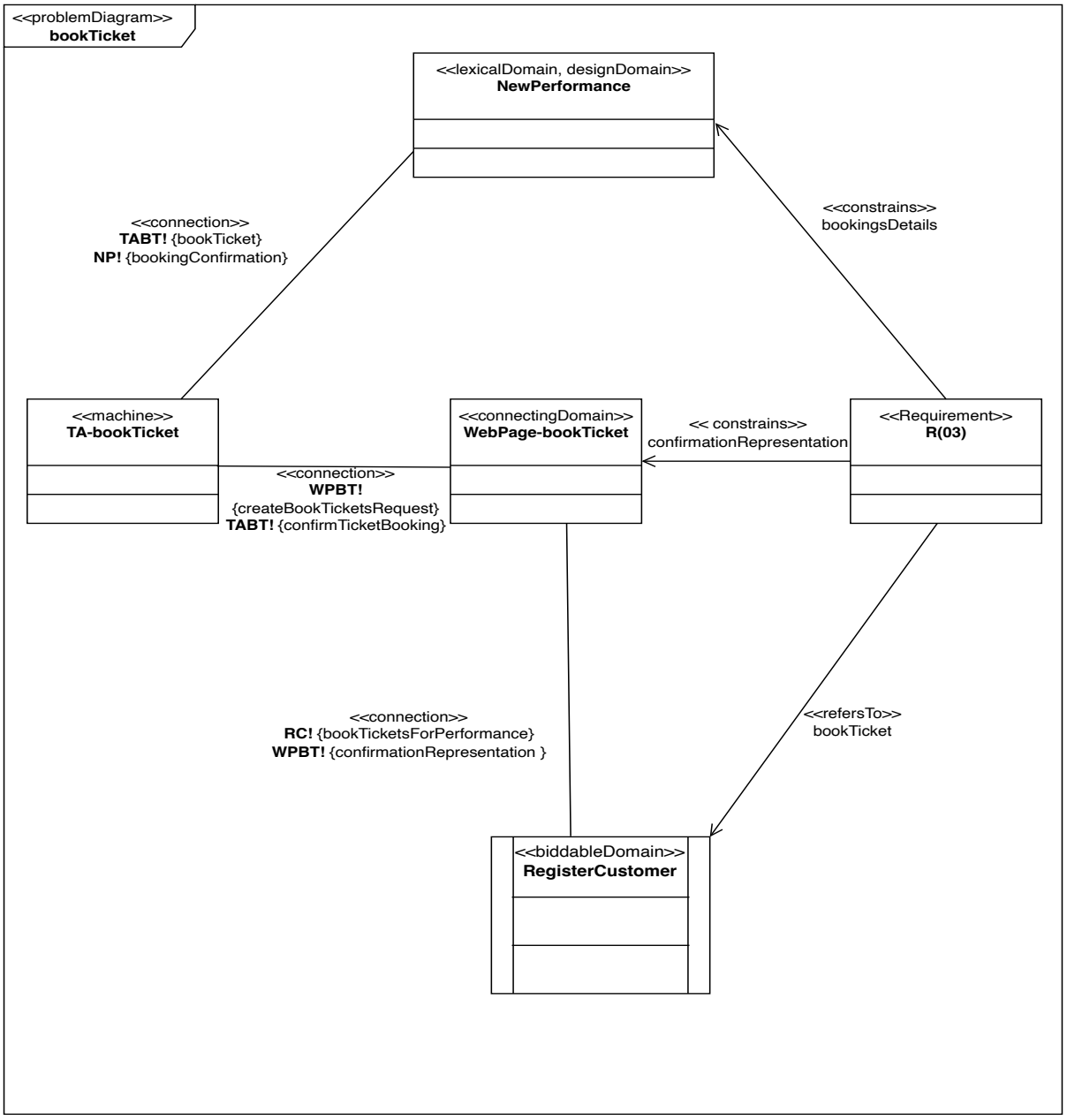


R2 Mapping

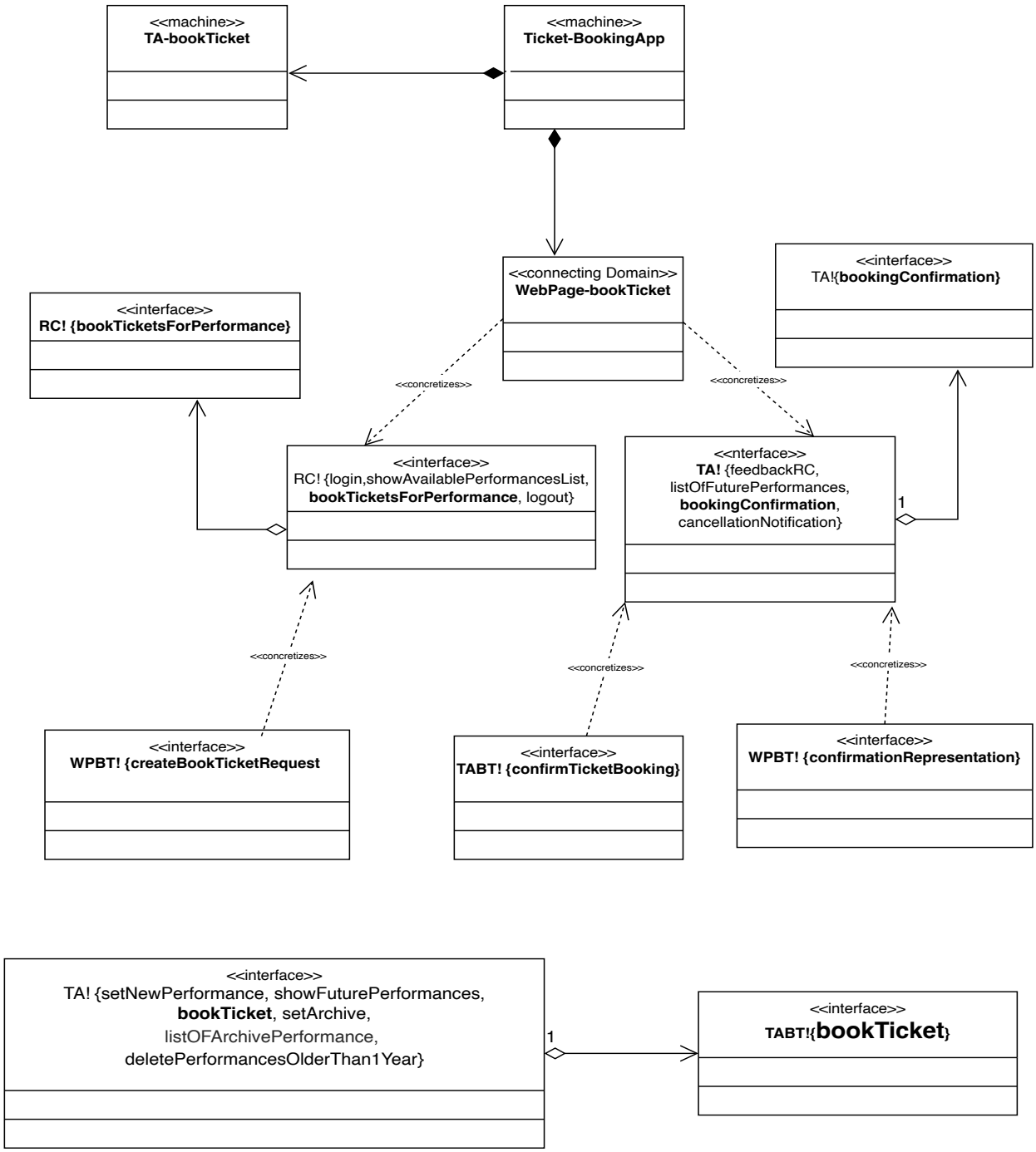

<<machine>
TA_List
<<machine>
Ticket-BookingApp
<<connectionDomain>>
WebpageOfFuturePerformances
<<concretizes>>
<<concretizes>>
<<interface>>
TA!{listOfFuturePerformances}
<<interface>>
TA! {feedbackRC, listOfFuturePerformances, bookingConfirmation, cancellationNotification}
1
<<interface>>
RC! {login, showAvailablePerformancesList, bookTicketsForPerformance, logout}
<<interface>>
RC!{showAvailablePerformancesList}
<<concretizes>>
<<concretizes>>
<<concretizes>>
<<interface>>
WOFP!{performanceListRepresentation}
<<interface>>
TAL!{sortedListOfFuturePerformance}
<<interface>>
WOFP!{showListOfFuturePerformances}
<<interface>>
TAL!{showFuturePerformances}
<<interface>>
TA!{ setNewPerformance,
showFuturePerformances,
book-Ticket, setArchive ,listOFArchivePerformance}
1

R3 Problem Diagram

Instantiation of Problem Frame “Update2”

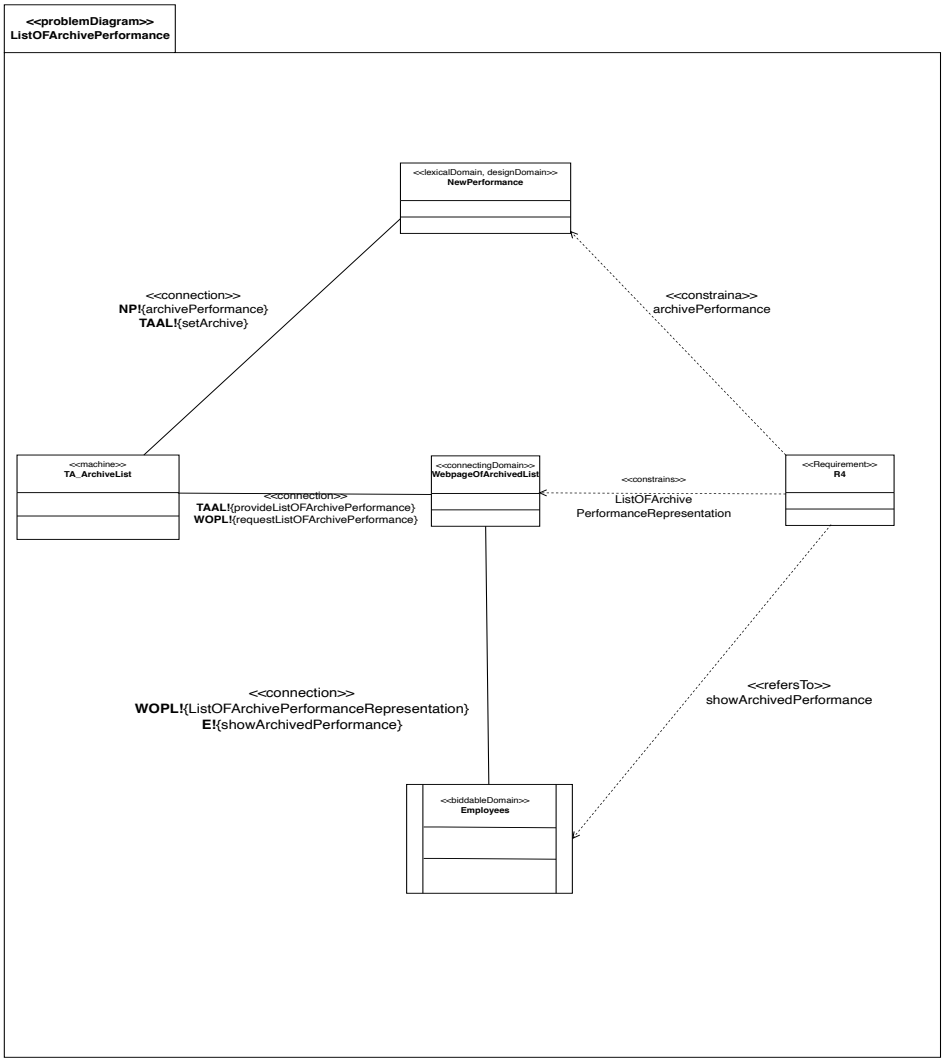


R3 Mapping

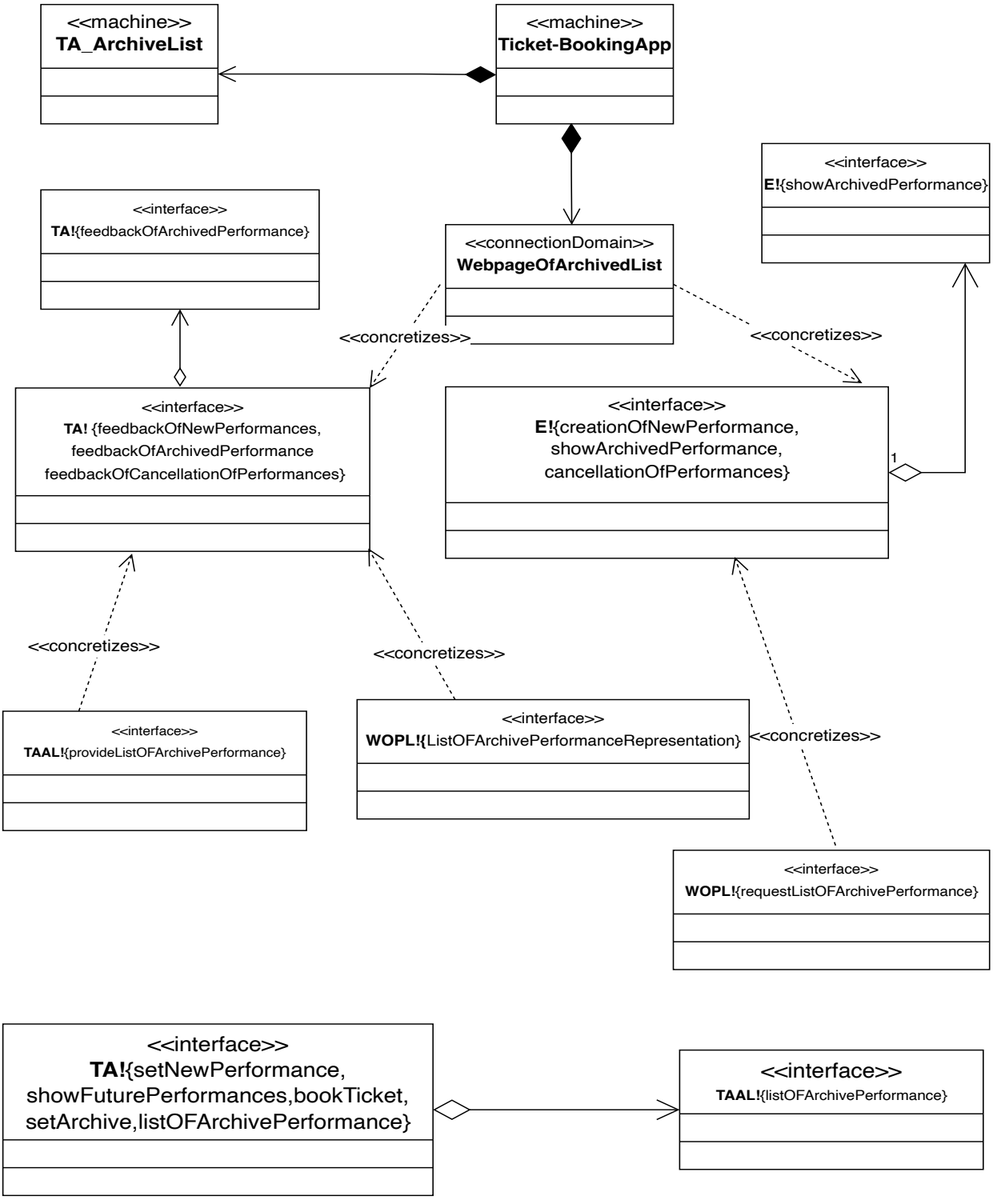

<<machine>>
TA-bookTicket
<<machine>>
Ticket-BookingApp
<<connecting Domain>>
WebPage-bookTicket
<<interface>>
TA!{bookingConfirmation}
<<interface>>
RC! {bookTicketsForPerformance}
<<concretizes>>
<<concretizes>>
<<interface>>
RC! {login,showAvailablePerformancesList,
bookTicketsForPerformance, logout}
<<nterface>>
TA! {feedbackRC,
listOfFuturePerformances,
bookingConfirmation,
cancellationNotification}
1
<<concretizes>>
<<concretizes>>
<<concretizes>>
<<interface>>
WPBT! {createBookTicketRequest
<<interface>>
TABT! {confirmTicketBooking}
<<interface>>
WPBT! {confirmationRepresentation}
<<interface>>
TA! {setNewPerformance, showFuturePerformances,
bookTicket, setArchive,
listOFArchivePerformance,
deletePerformancesOlderThan1Year}
1
<<interface>>
TABT!{bookTicket}

R4 Problem Diagram

Instantiation of Problem Frame “Query2”



R4 Mapping


<<machine>>
TA_ArchiveList
<<machine>>
Ticket-BookingApp
<<interface>>
E!{showArchivedPerformance}
<<interface>>
TA!{feedbackOfArchivedPerformance}
<<connectionDomain>>
WebpageOfArchivedList
<<concretizes>>
<<concretizes>>
<<interface>>
TA! {feedbackOfNewPerformances,
feedbackOfArchivedPerformance
feedbackOfCancellationOfPerformances}
<<interface>>
E!{creationOfNewPerformance,
showArchivedPerformance,
cancellationOfPerformances}
1
<<concretizes>>
<<concretizes>>
<<interface>>
TAAL!{provideListOFArchivePerformance}
<<interface>>
WOPL!{ListOFArchivePerformanceRepresentation}
<<concretizes>>
<<interface>>
WOPL!{requestListOFArchivePerformance}
<<interface>>
TA!{setNewPerformance,
showFuturePerformances,bookTicket,
setArchive,listOFArchivePerformance}
<<interface>>
TAAL!{listOFArchivePerformance}

A3 (Sequence Diagram)

Specification of R01

Using the domain knowledge

(A2): Only self-accessible e-mails will be provided by the customers.

we can derive the specifications:R01

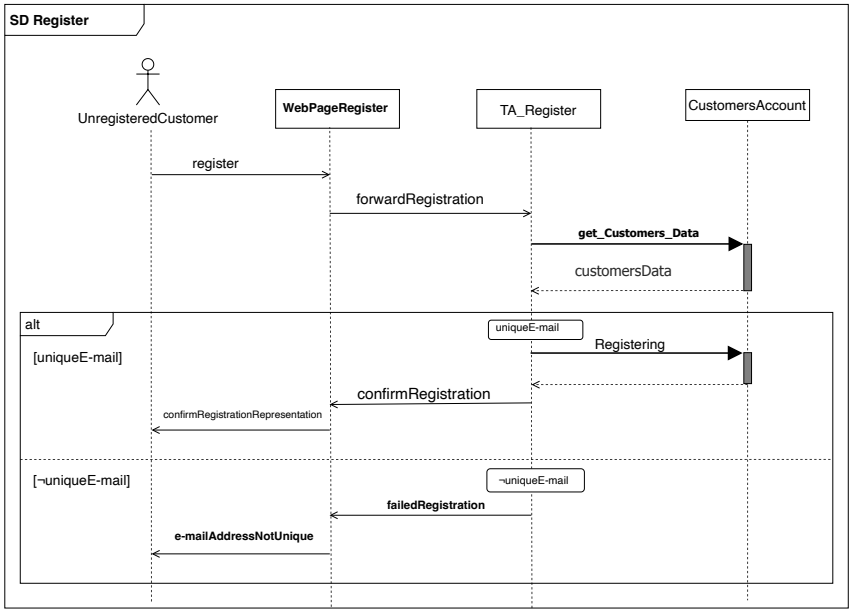
WebpageRegister (S01a): When the webpage receives the command "register", then the command is forwarded to the machine with the command "forwardRegistration". The feedback is received via the command "confirmaRegistration". As Feedback "confirmRegistrationRepresentation" is shown to the UnregisterCustomer .

TA_Register(S01b): When the machine receives the command "forwardRegistration" from Web Page , Machine itself gives another command named "get_Customers_Data" to the Lexical Domain – CustomersAccount to get customers data (email address). Then CustomersAccount (Lexical Domain) provides Machine the needed customers details with the command "customerData". After having the details of customers from the lexical domain Machine will check if it was used before or not. It's mainly to maintain the unique Email addresses for every single account. If it is the case then Machine will send another command to Lexical Domain named "Registering" for the further registration process. And also Machine will confirm Web Page about the registration Confirmation with the command "confirmRegistration". And if the given email address is used before then machine will notify the unregistered customer about the registration failure through Web Page through this command "failedRegistration".

CustomerAccount (S01c): After receiving the command "get_Customers_Data" from the Machine CustomersAccount will provide all registered customers data to Machine at first. Then following the customers details CustomersAccount (Lexical Domain) have possibility to get one more order named "Registering" and will save the data.

Correctness condition: $(S01a) \wedge (S01b) \wedge (S01c) \wedge (A02) \Longrightarrow (R01)$

Sequence diagram R01



Specification of R02
Using the domain knowledge

(A03): Customers will check this page regularly to find out new ideas.

we can derive the specifications:R02

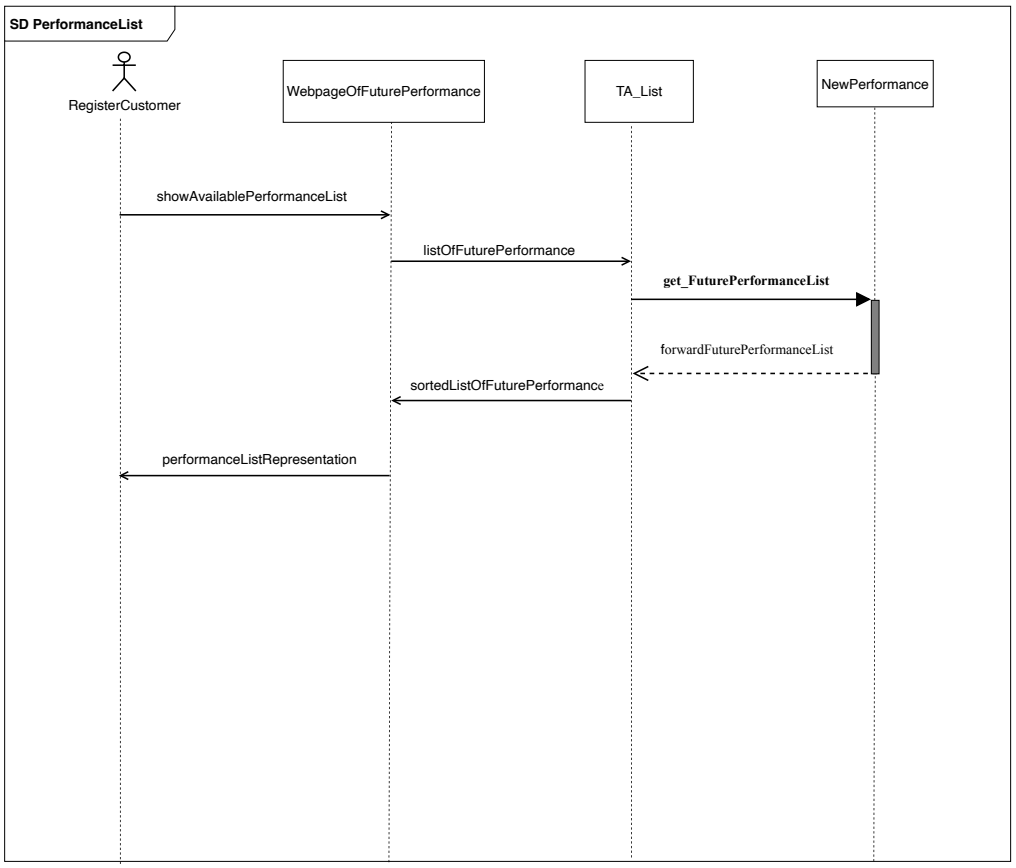
WebpageOfFuturePerformance (S02a): When the webpage receives the command "showAvailablePerformancesList" than the command is forwarded to the machine with the command "showListOfFuturePerformance". The results are received via the command "sortedListOfFuturePerformance" and shown to the RegisterCustomer by "PerformanceListRepresentation".

TA_List (S02b): When the machine receives the command "showListOfFuturePerformance", the performance list is selected with the command "get_FuturePerformanceList" and received as the data " forwarFuturePerformanceList ". The result is returned via the command "sortedListOfFuturePerformance".

NewPerformance (S02c): After receiving the command "get_FuturePerformanceList" the results are returned as the data "forwardFuturePerformanceList".

Correctness condition: (S02a) ∧ (S02b) ∧ (S02c) ∧ (A03) ⟹ (R02)

Sequence Diagram of R02



Specification of R03

Using the domain knowledge

(F01): A performance consists of the title of the film, the duration, an assigned hall and a date with time on which the film will be shown.

(F02): A hall consists of a predetermined number of rows, the number of seats per row and a unique hall number.

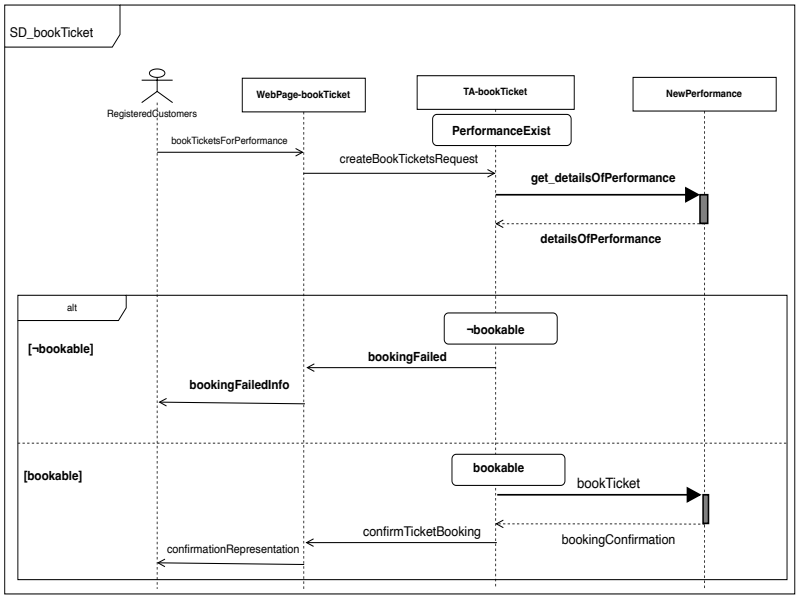
we can derive the specifications:R03

WebPage-bookTicket (S03a) : When the webpage receives the command "bookTicketsForPerformance" than the command is forwarded to the machine with the command "createBookTicketsRequest". The results are received via the command "confirmTicketBookingand shown to the RegisteredCustomer by "confirmationRepresentation"

TA-bookTicket (S03b) : When the machine receives the command "createBookTicketsRequest" then the Machine send a command to NewPerformance which is named as "get_detailsOfPerformance" to get the specific performance data. As a result NewPerformance or the Lexical Domain also provide the specific performance details to the Machine with the following command named as "detailsOfPerformance". After getting the specific performance details from NewPerformance the Machine will calculate the time itself and if it is having more than 15 minitues to start the performance then Machine will send another command to New Performance for booking a ticket which is named as "bookTicket". On the contrary New Performance will also give a booking confirmation with this command named as "bookingConfirmation" in this case. But if it is having less than 15 minitues of time for a performance to be started then Machine will send the registered customer's web page a booking failure with the command "bookingFailed".

NewPerformance (S03c) : After receiving the command "get_detailsOfPerformance" from Machine, New Performance will provide specific performance details to Machine with the command "detailsOfPeformance". Not only that but also New performance will Send booking confirmation to Machine with the command "bookingConfirmation" as the answer to the command "bookTicket"from Machine.

Correctness condition: $(S03a) \wedge (S03b) \wedge (S03c) \wedge (F01) \wedge (F02) \implies (R03)$


SD_bookTicket
RegisteredCustomers
WebPage-bookTicket
TA-bookTicket
NewPerformance
PerformanceExist
bookTicketsForPerformance
createBookTicketsRequest
get_detailsOfPerformance
detailsOfPerformance
alt
¬bookable
[¬bookable]
bookingFailed
bookingFailedInfo
bookable
[bookable]
bookTicket
confirmTicketBooking
bookingConfirmation
confirmationRepresentation

Specification of R04

we can derive the specifications:R04

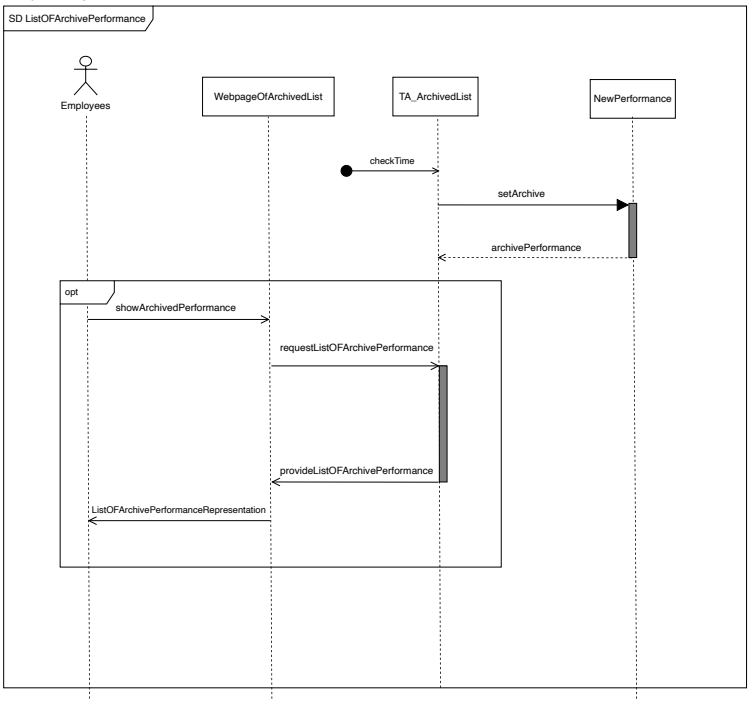
WebPageOfArchiveList(S04a): When the webpage receives the command "showArchivePerformace", then the command is forwarded to the machine with the command "resquestListOfArchivePerformance". The results are received via the command "providetListOfArchivePerformance" and shown to the Employees by "listOfArchivePerformaceRepresentation"

TA_ArchiveList(S04b): When the machine receives the command "resquestListOfArchivePerformance" from the Web Page (Connection Domain), then it provides The Archive performance details to the Web Page with the command "provideListOfArchivePerformance". In the mean time before providing the Archive Performance details to Web Page Machine send a Command to New Performance named as "setArchive" to set the performances as archive according to the requirements and conditions and whenever employees would like to see them New Performance will provide the Archive details to Machine with the command "archivePerformance".

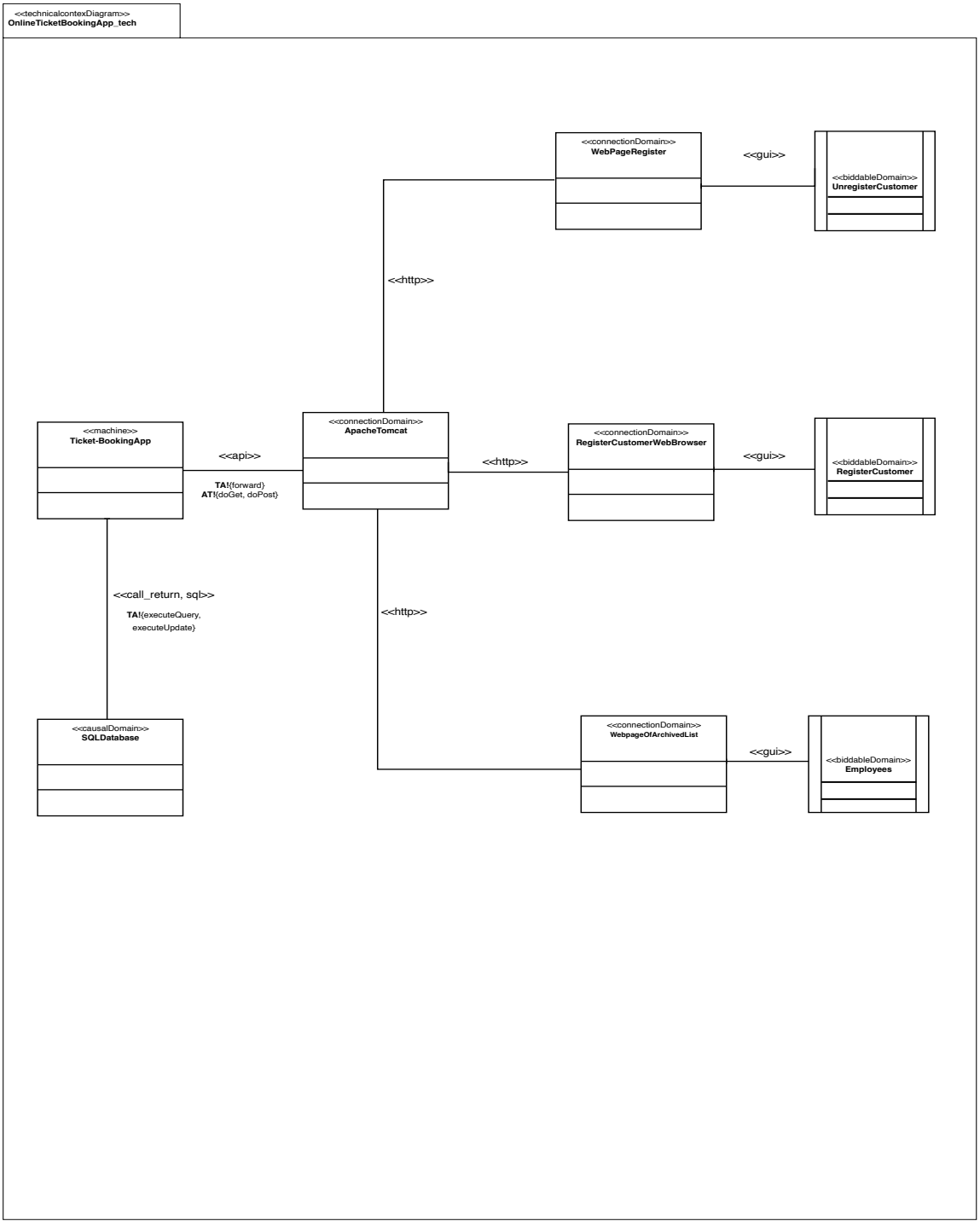
NewPerformance(S04c): After receiving the command "setArchive" the results are returned as the data "archivePerformance".

Correctness condition: (S04a) ∧ (S04b) ∧ (S04c) ⟹ (R04)

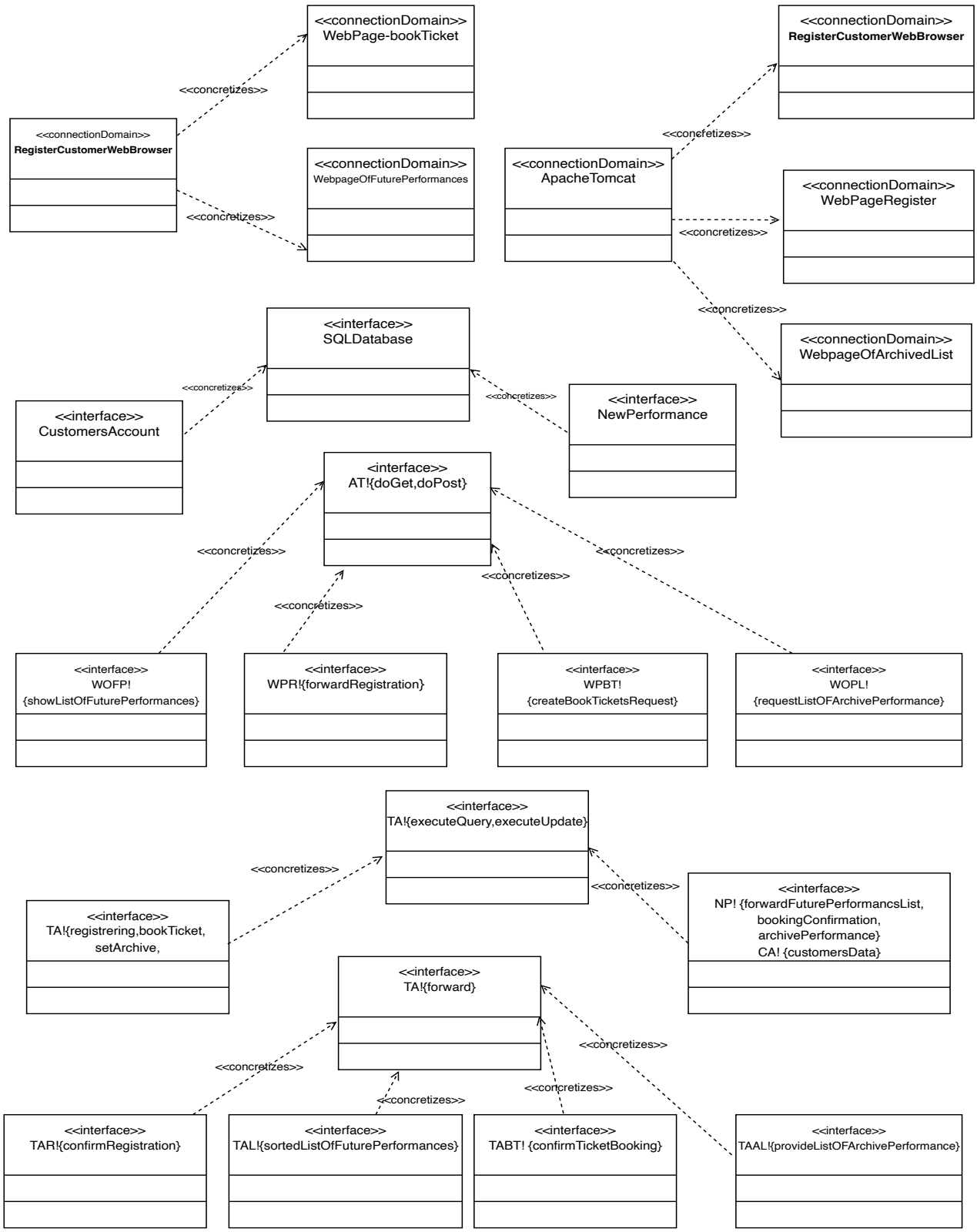
Sequence Diagram of R04



A4 (TechnicalContext Diagram)



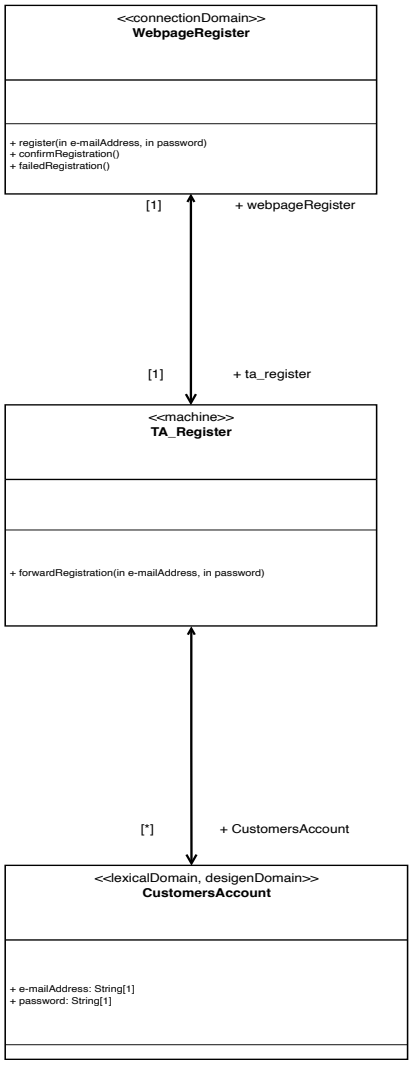
TechnicalContext Daigram mapping



A5 Class Model and Operations Specifications

R01: The operation Register

class model:



Class Diagram R1
Name:forwardRegistration
Description: Try to register customer and return confirm or faile
OCL Constrain:

```
context TA_Register::forwardRegistration(e-mailAddress:
String, password: String)
pre: true
post: if not  CustomerAccount@pre->one (c: customerAccount
| c.e-mailAddress = e-mailAddress)
then
CustomerAccount -> one( c: customerAccount | c.e-
mailAddress = e-mailAddress and c. password = password )
and customersAccount->size() = customersAccount @pre-
>size() + 1 and

 webpageRegister^confirmRegistration()

Else
webpageRegister^failedRegistration()

endif
```

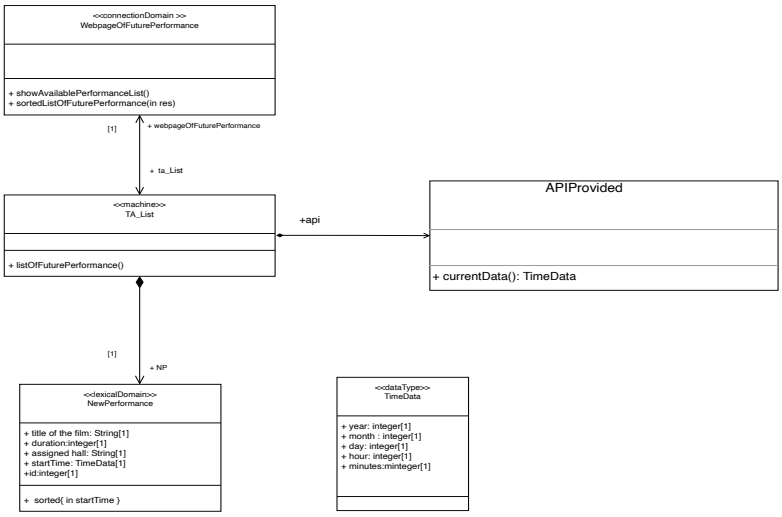
Remark: We want to be able to identify customer by a unique e-mailAddress.

OCL constraint:

```
context customer
inv : CustomerAccount . allInstances()->isUnique(e-
mailAddress)
```

Remark: To be clearly identifiable an e-mail address can only be used exactly by one customer(R01)

R02: The operation TA_List



Name: showAvailablePerformanceList
Description: Forwards the show request from the Register Customer to the machine.
OCL Constraint:

Context: WebpageOfFuturePerformance :: showAvailablePerformanceList ( )
Pre true
Post ta_List^listOfFuturePerformance()

Name: listOfFuturePerformance
Description : Checks the specified date is greater than date of the performance
OCL constraint:

Context TA_List :: listOfFuturePerformance ()
Pre : true
Post: let res: set (NewPerformance) = NP->select(np:NewPerformance | np.startTime > api.currentTime) ->asSet()

In NP->sortedBy(np:NewPerformance | np.startTime)

And WebpageOfFuturePerformance^ sortedListOfFuturePerformance(res)

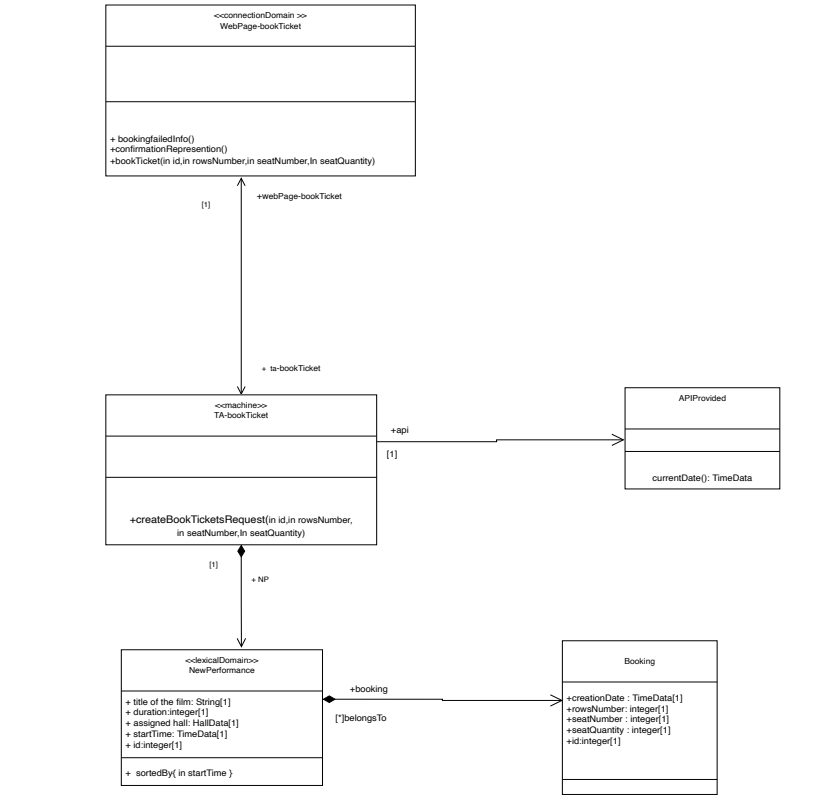
Remark: We want to be able to identify newPerformance by a unique id.

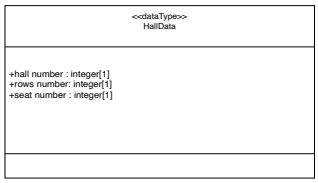
OCL constraint:

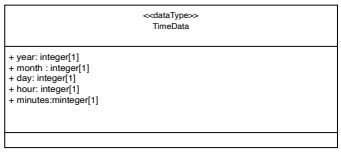
Context newPerformance

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

R03: The operation bookTicketsForPerformance







Name: Ticket-Booking

Description: book ticket for a performances and given data

OCL Constraint:

```
Context WebPage-bookTicket:: bookTicket
                              (id:integer , seatNumber:integer,
                              seatQuantity:integer, rowsNumber:integer)
pre : true


post ta_bookTicket^createBookTicketsRequest(id,rowsNumber,seatNumber, seatQuantity)
```

Name: create book Ticket request

Description: book ticket for a performances and returns confirm or failed booking

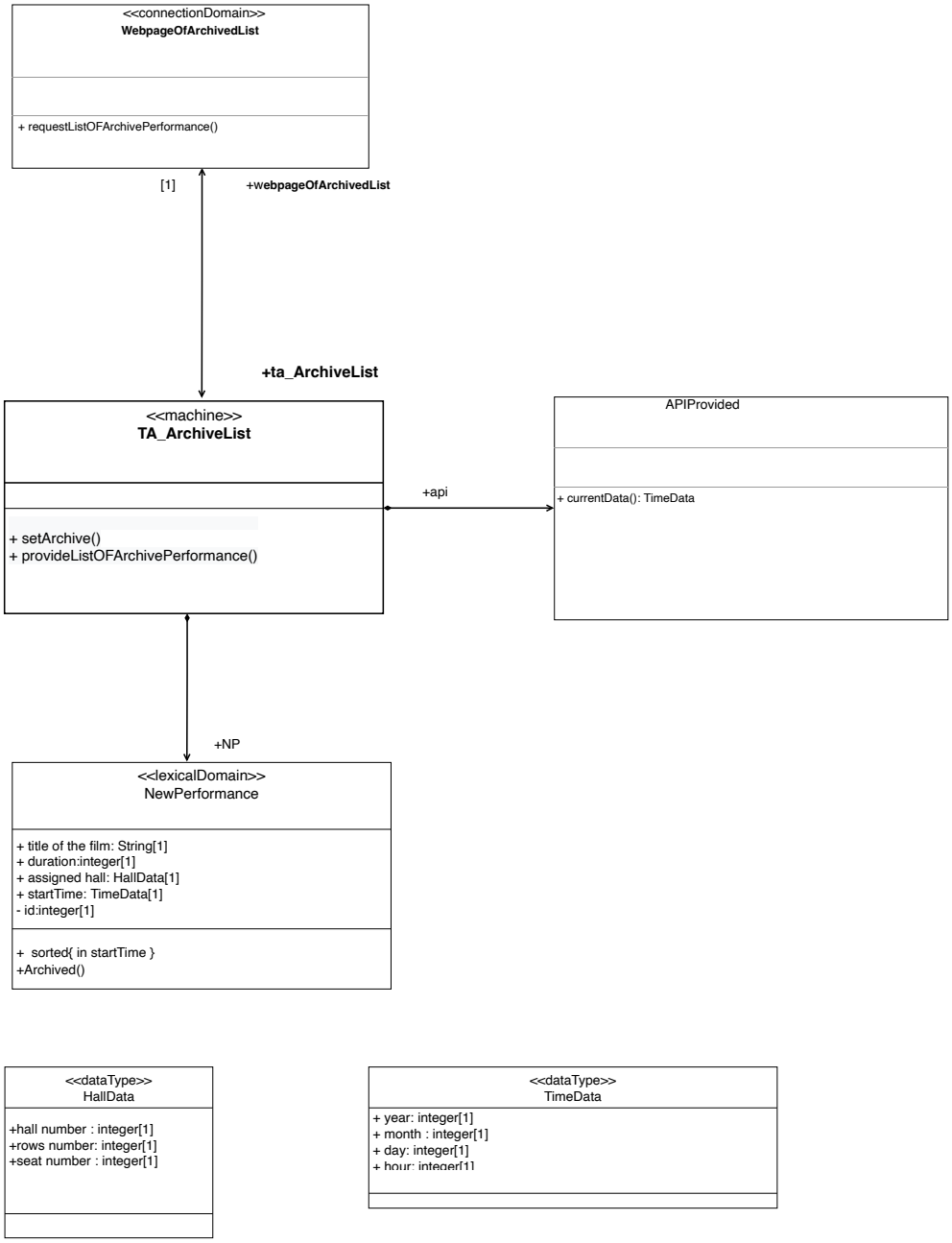
OCL Constraint:

```
Context TA_bookTicket:: bookTicket
                              (id:integer , seatNumber:integer,
                              seatQuantity:integer, rowsNumber:integer)
pre : NP->one(np.NewPerformance | np.id = id)


post let np:NewPerformance = NP->any(np:NewPerformance | np.id = id) in
if ( np.startTime-15min)>api.currentTime() then
np.booking->one(b:Booking | b.creationData = api.currentDate() and b.rowsNumber = rowsNumber
and b. seatNumber = seatNumber and b. seatQuantity= seatQuantity)

np.booking->size() = np.booking@pre->size ()+1 and

webpage-bookTicket^confirmationRepresention()

else
webpage-bookTicket^bookingfailedInfo()
```

Booking have a unique id
OCL constraint:

```
Context booking
inv :  booking.allInstances()->isUnique( id )
```

R04: The operation ListOFArchivePerformance



Name: setArchive
Description: when performances will be start then machine set performance as archive automatically .
OCL Constraint:

```
Context TA_ArchiveList :: setArchive

pre true
post: let np:NewPerformance = NP->select(np:NewPerformance | np.startTime= api.currentData())
in
      if NP@pre.startTime =  api.currentData()
Then
      np.Archived()
     endif
```

Name:requestListOFArchivePerformance
Description: on request of archive performance from employees the machine show the archive performance list
OCL Constraint:

```
Context TA_ArchiveList :: requestListOFArchivePerformance

pre : true
post: if AL@pre.startTime <= api.currenetDate()  Then
WebpageOfArchivedList ^  provideListOFArchivePerformance()

endif
```

newperformance have a unique id
OCL constraint:

```
Context : TA_ArchiveList
inv : NewPerformance.allInstances()->isUnique( id )
```

A6 Software lifecycle

$LC_{UnregisteredCustomers}$=(Register ; [PerformanceList]* ;[bookTicket]*)

$LC_{RegisteredCustomers}$=([PerformanceList] ;[bookTicket])*

$$LC_{online\_Ticket\_booking\_app} = (||^{n}_{i=1} LC_{UnregisteredCustomers_i}) \; || \; (||^{n}_{j=1} LC_{RegisteredCustomers_i}) || \; ListOFArchivePerformance^*$$

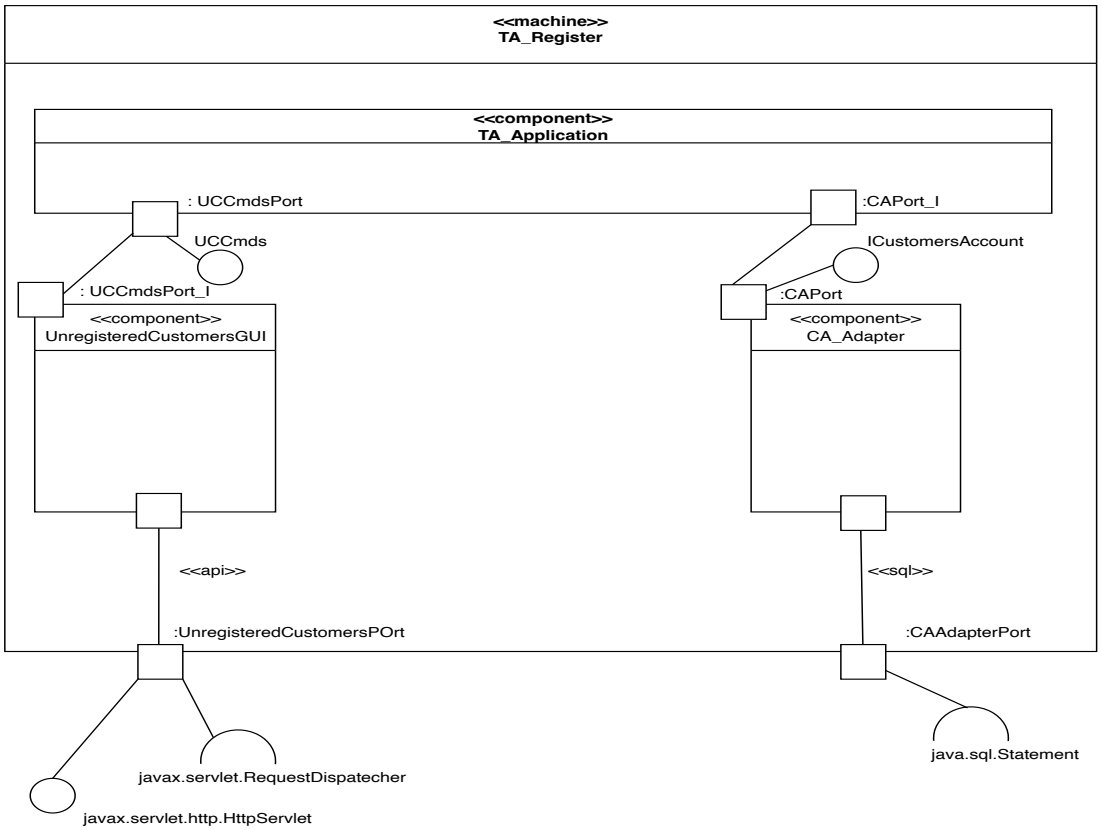
where $||^{n}_{i=1} LC_i$ denotes the parallel composition of n copies of

life-cycle LC.

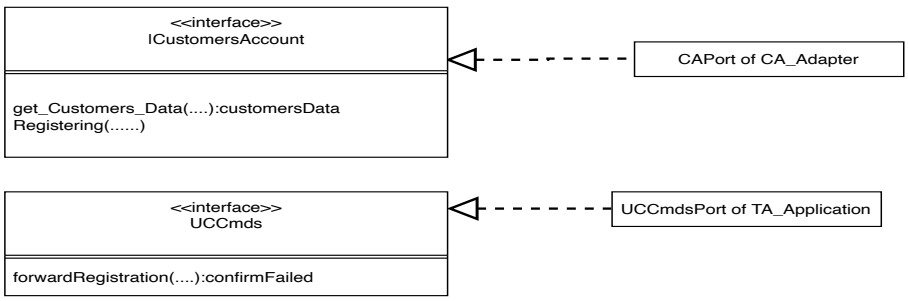
D1

Online Ticket-Booking App software architecture:

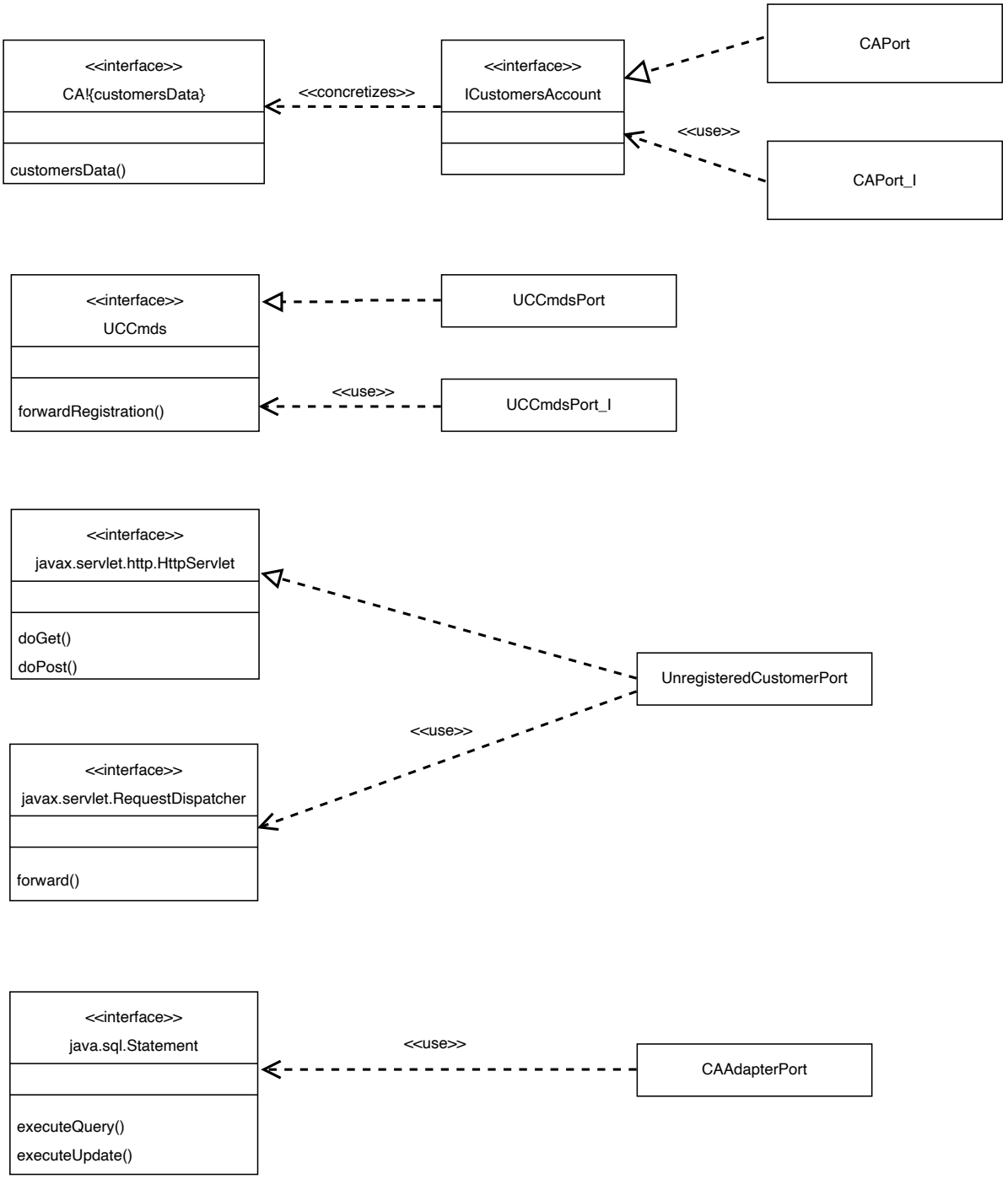
TA_Register fits to Update (2). Instantiated architectural pattern for TA_Register:



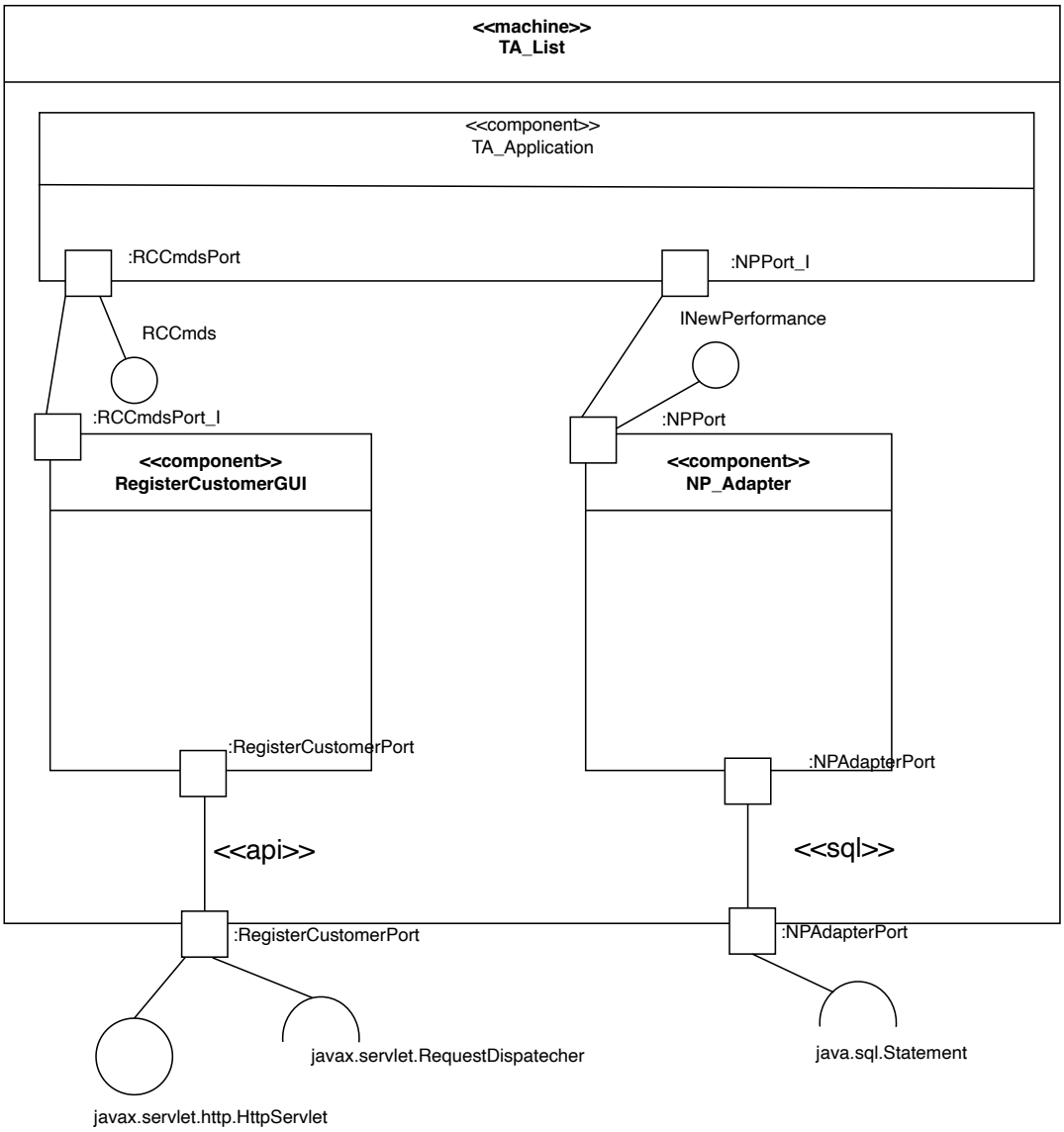
Internal interfaces in TA_Register:



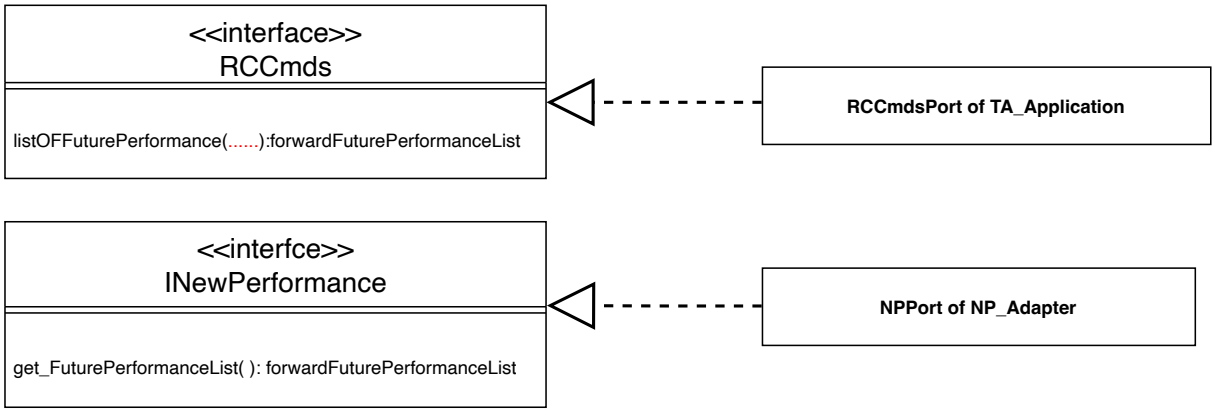
port types and interface relations for TA_Register



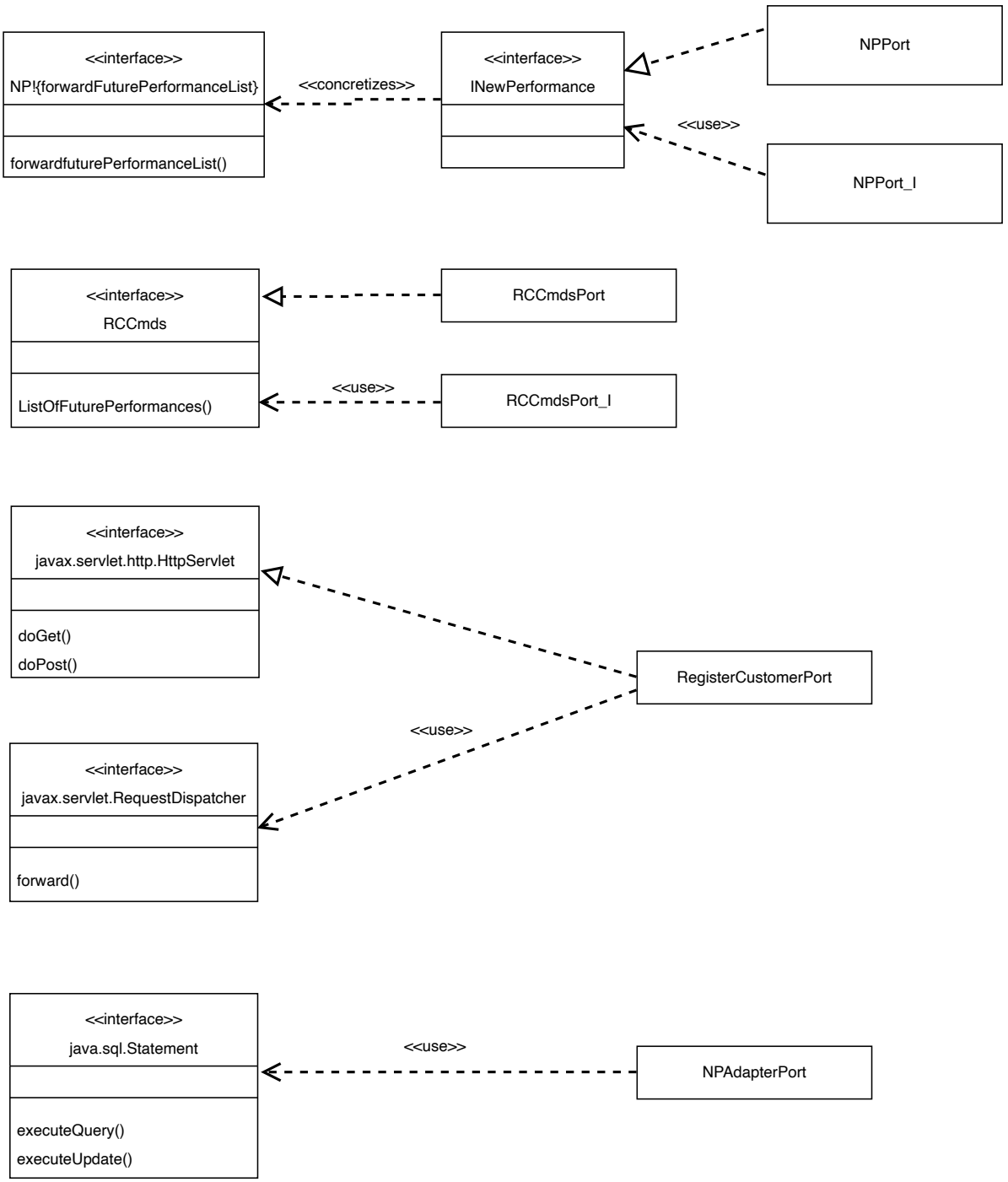
TA_List fits to query (2). Instantiated architectural pattern for TA List:



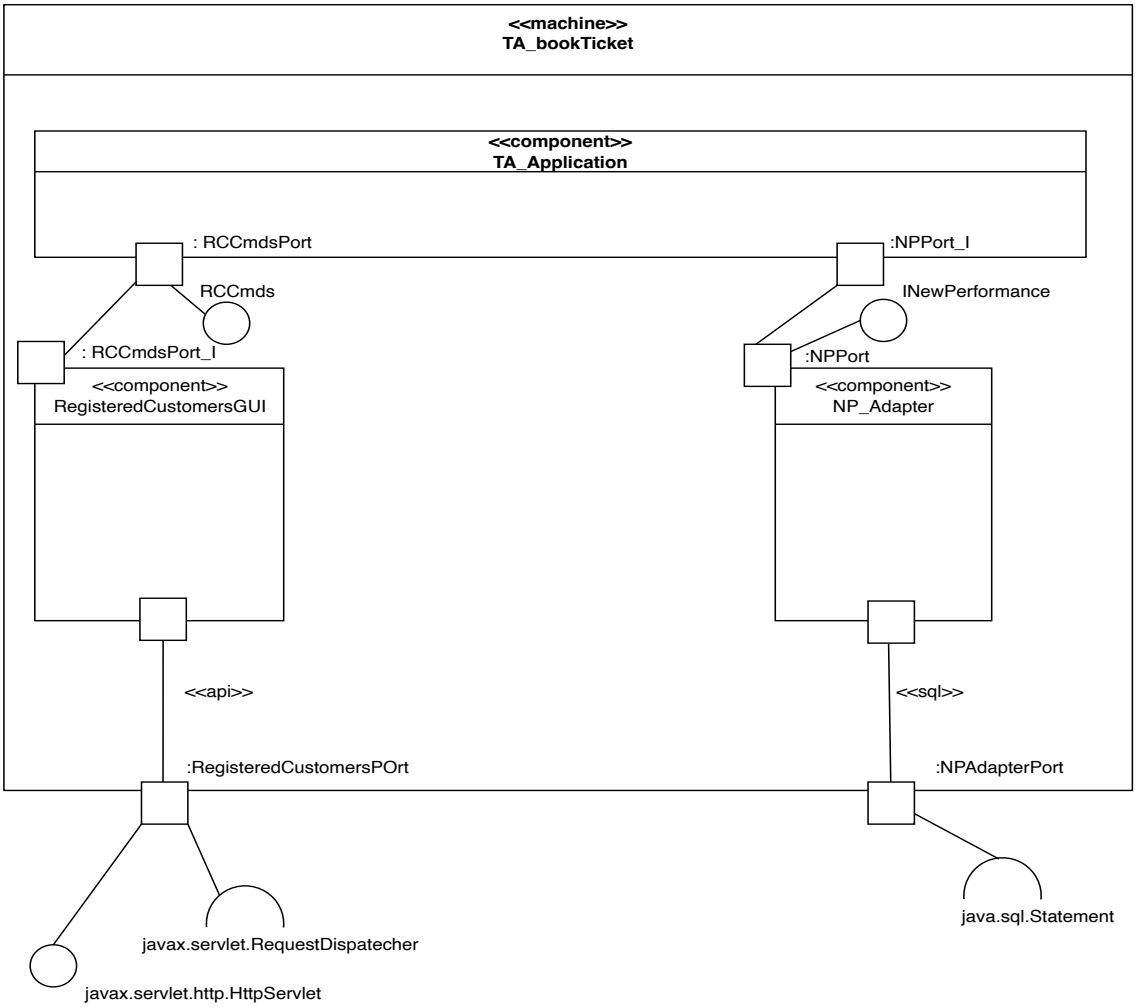
Internal interfaces in TA_List:



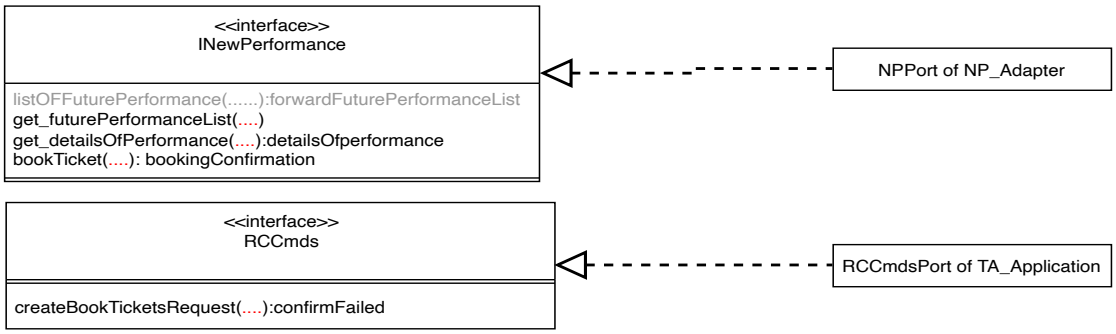
Port types and interface relations for TA_List



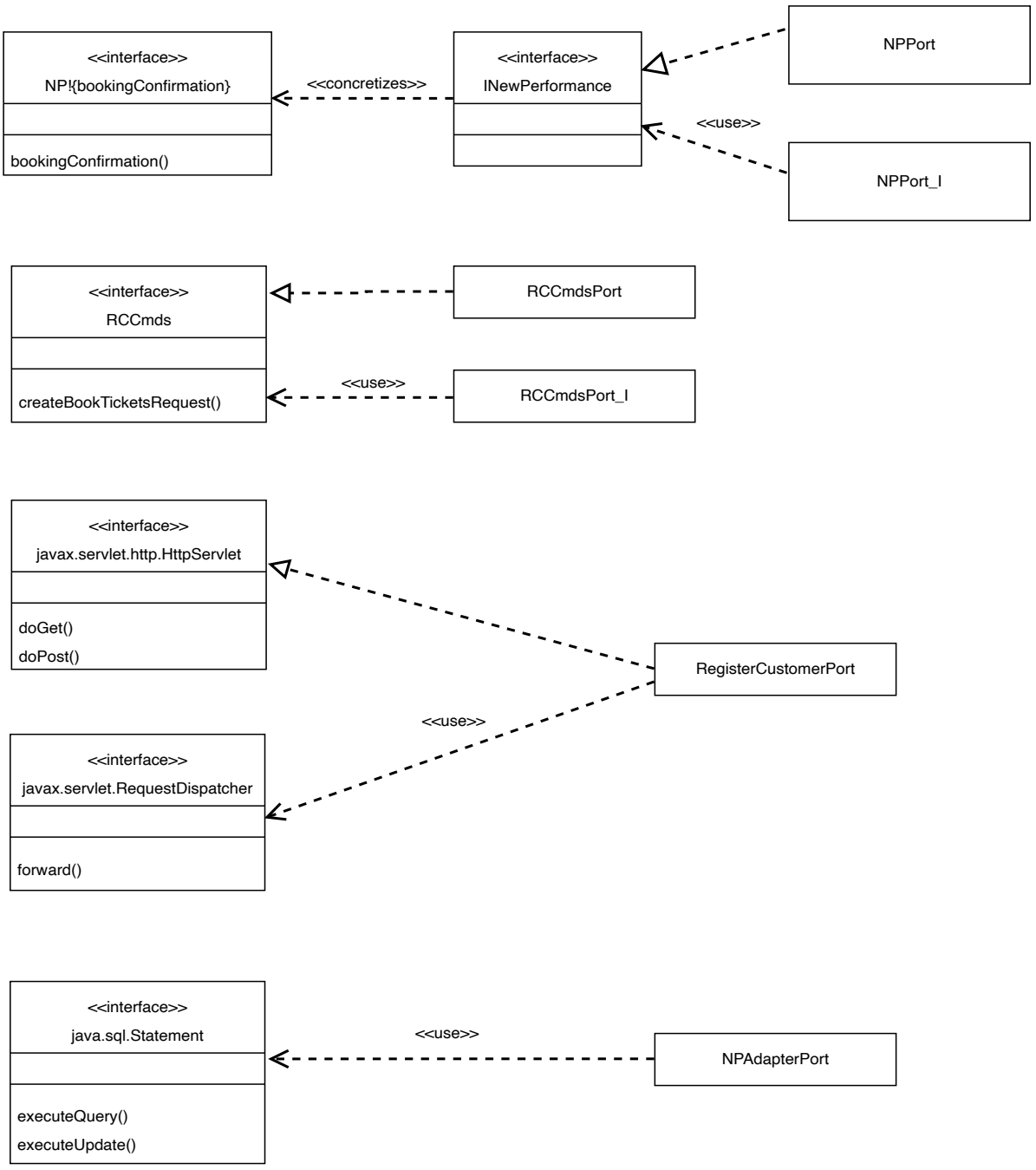
TA_bookTicket fits to Update (2). Instantiated architectural pattern for TA _bookTicket



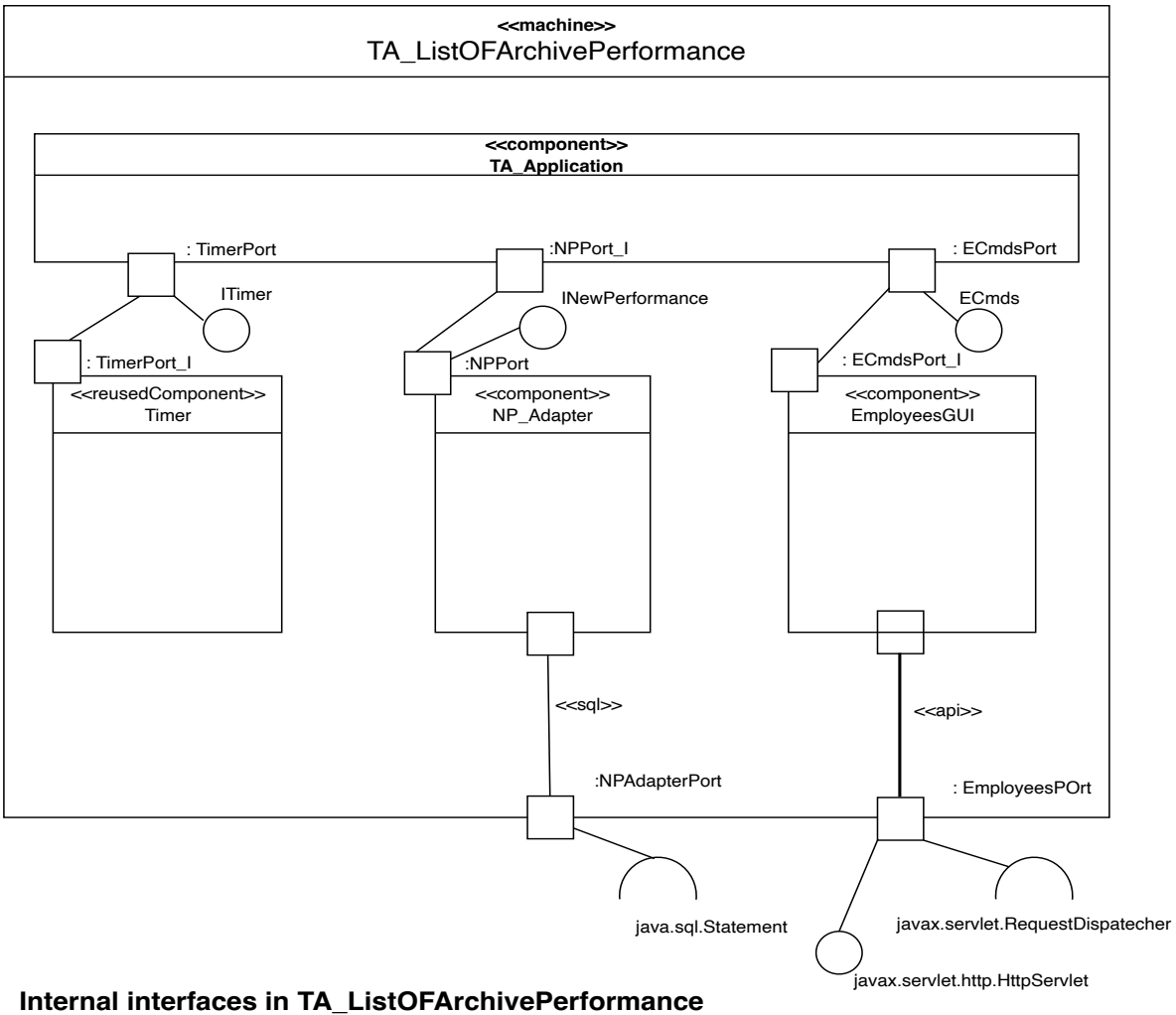
Internal interfaces in TA_bookTicket:



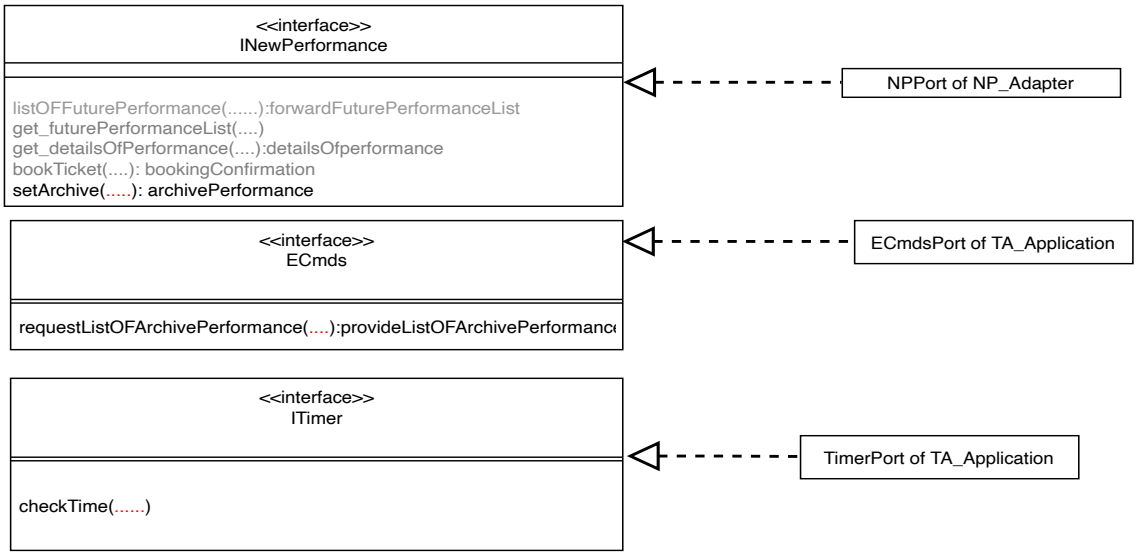
port types and interface relations for TA_bookTicket



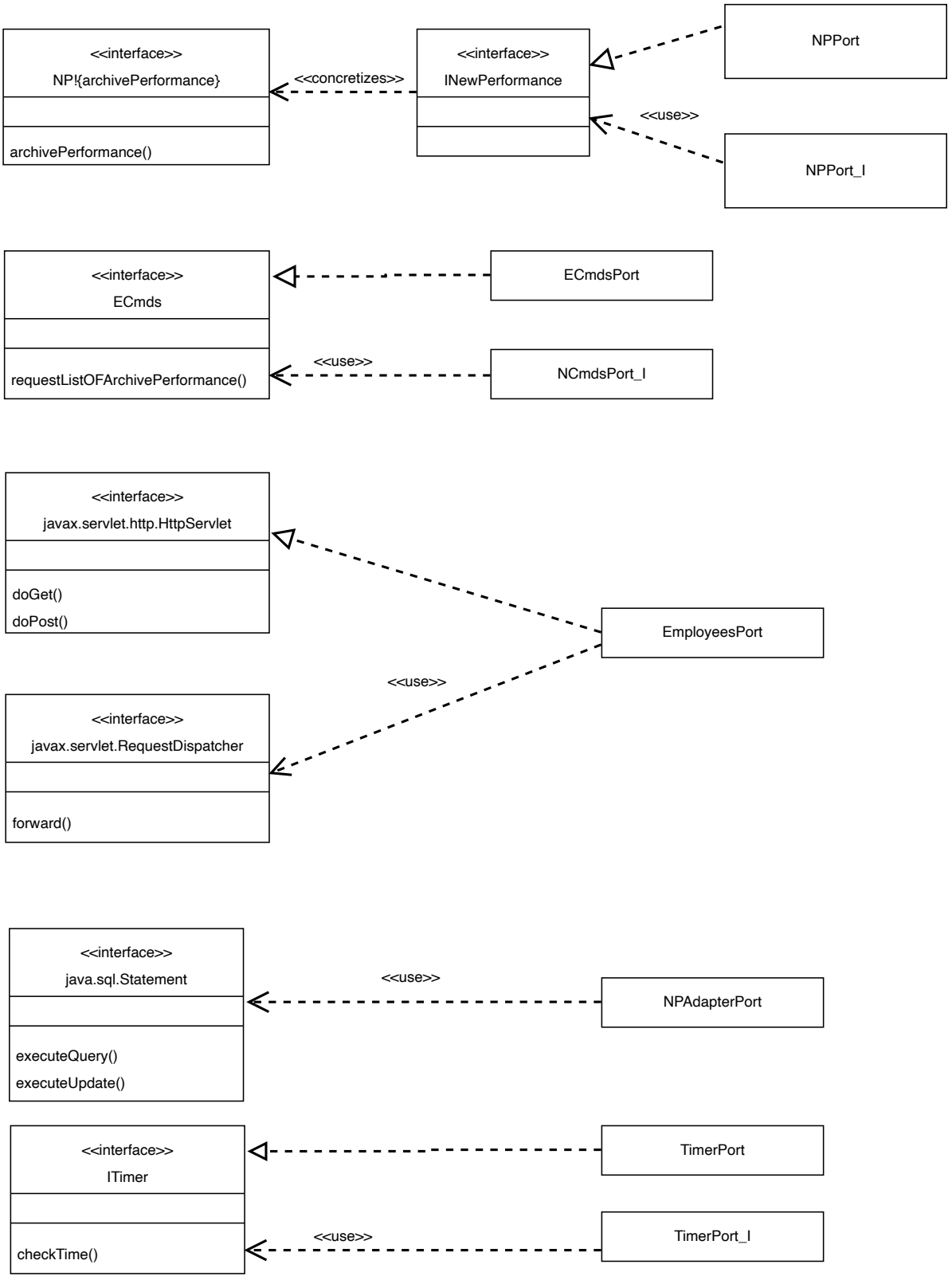
TA_ListOFArchivePerformance fits to query (2). Instantiated architectural pattern for TA_ListOFArchivePerformance :



Internal interfaces in TA_ListOFArchivePerformance

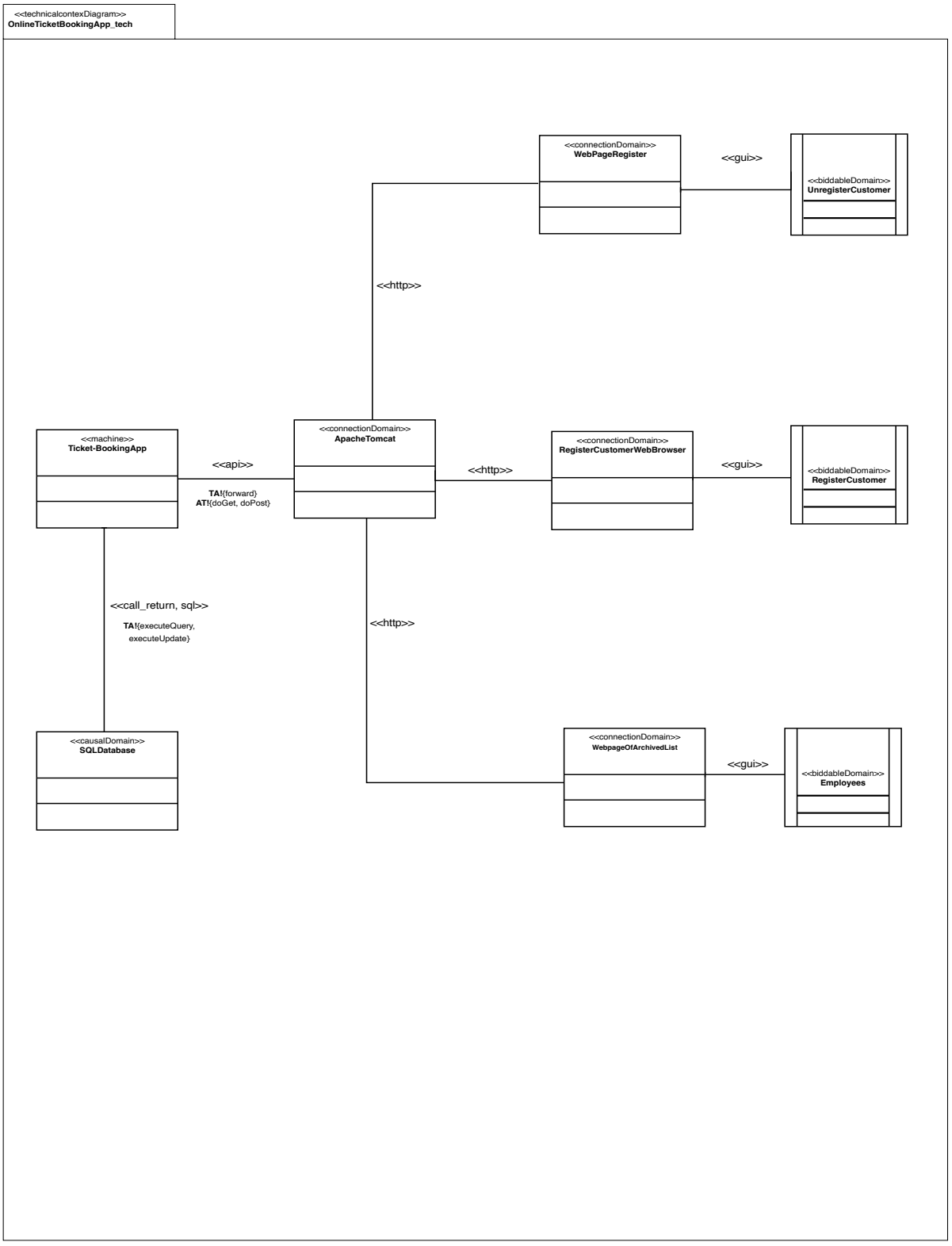


port types and interface relations for TA_ListOFArchivePerformance



Refining app if interface classes

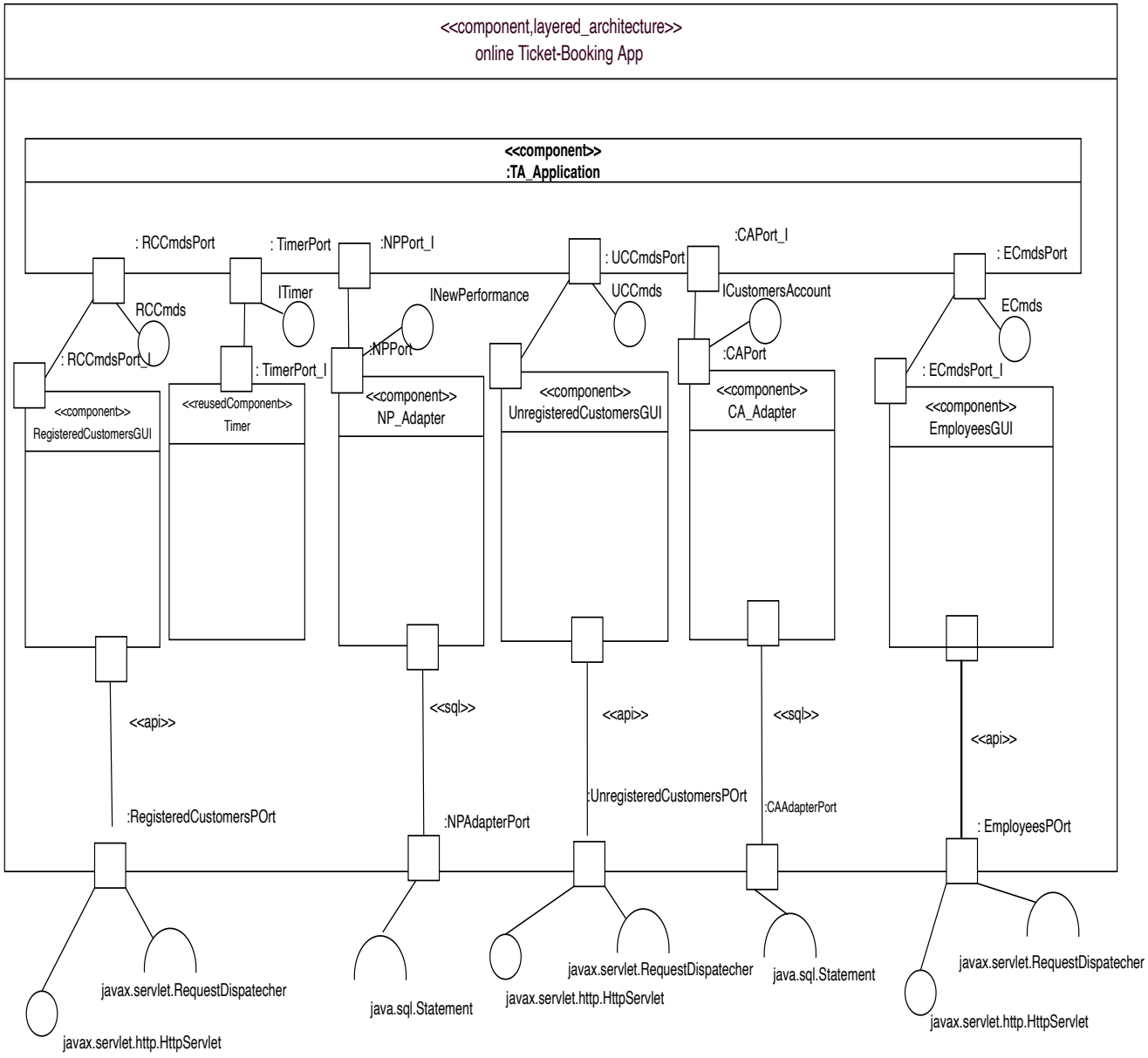
Refining tech if" interface classes



Refining “tech_if” interface classes

Considered interface in subproblem architecture	technical interface
<<api>> javax.servlet.http.HttpServlet in TA_List	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.RequestDispatcher in TA_List	<<api>> TA!{forward}
<<sql>> java.sql.Statement in TA_List	<<call return, sql>> TA!{executeQuery, executeUpdate}
<<api>> javax.servlet.http.HttpServlet in TA_Register	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.RequestDispatcher in TA_Register	<<api>> TA!{forward}
<<sql>> java.sql.Statement in TA_Register	<<call return, sql>> TA!{executeQuery, executeUpdate}
<<api>> javax.servlet.http.HttpServlet in TA_bookTicket	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.RequestDispatcher in TA_bookTicket	<<api>> TA!{forward}
<<sql>> java.sql.Statement in TA_bookTicket	<<call return, sql>> TA!{executeQuery, executeUpdate}
<<api>> javax.servlet.http.HttpServlet in TA_ListOFArchivePerformance	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.RequestDispatcher in TA_ListOFArchivePerformance	<<api>> TA!{forward}
<<sql>> java.sql.Statement in TA_ListOFArchivePerformance	<<call return, sql>> TA!{executeQuery, executeUpdate}

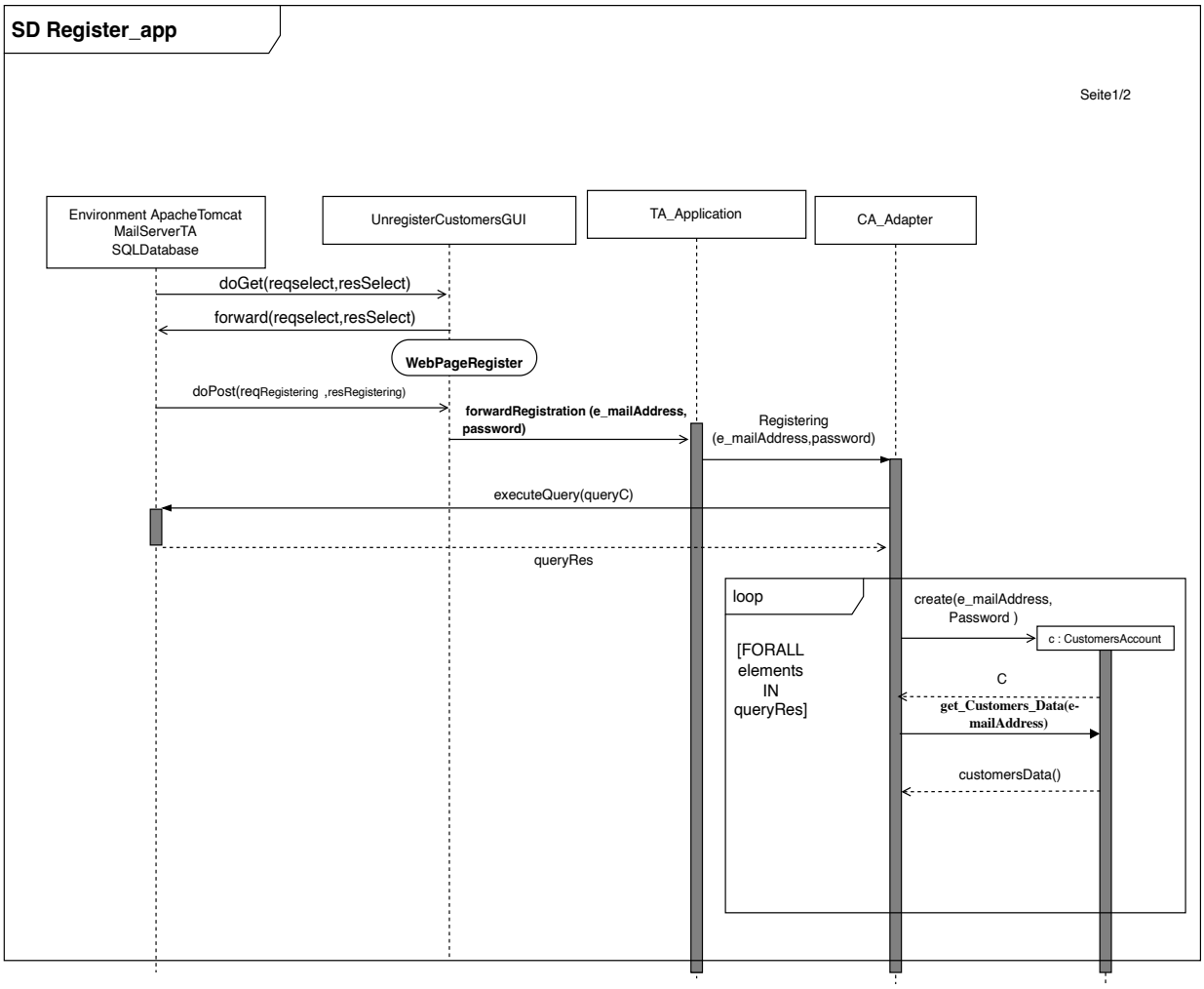
Global Architecture

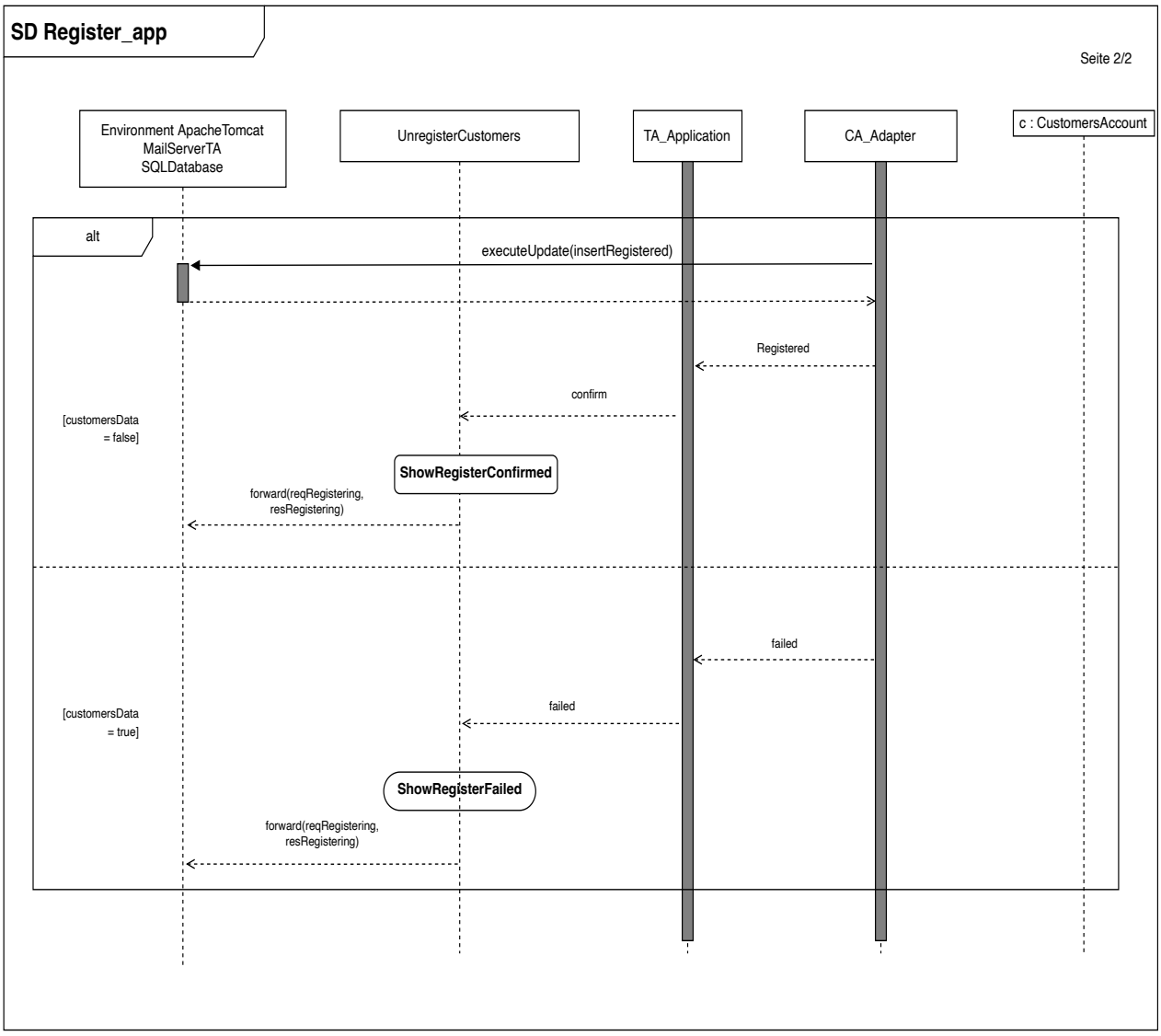


Inter-Component Interaction

- In our example, we will use MySQL as a database system.
- This relational database system requires a server software, i.e. MySQL server.
- We do not consider transactions or other commitment strategies for the presented implementation in this lecture.

Inter-component Interaction for R1 :





queryC

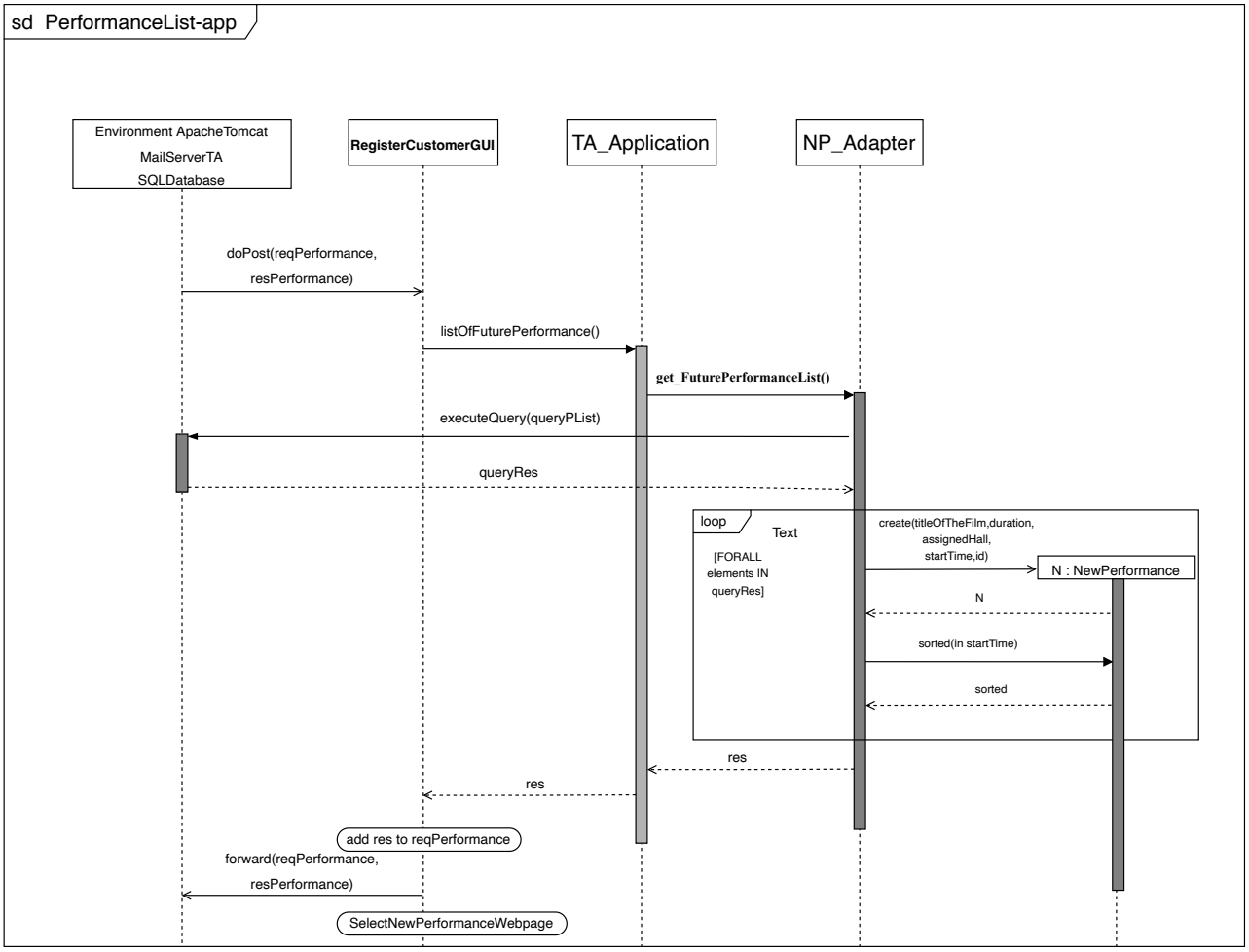
```
SELECT * FROM customersAccount WHERE e_mailAddress="e_mailAddress"
```

insertRegistering

```
INSERT INTO customersAccount (e_mailAddress, Password )
```

```
VALUES ("e_mailAddress"," Password" )
```

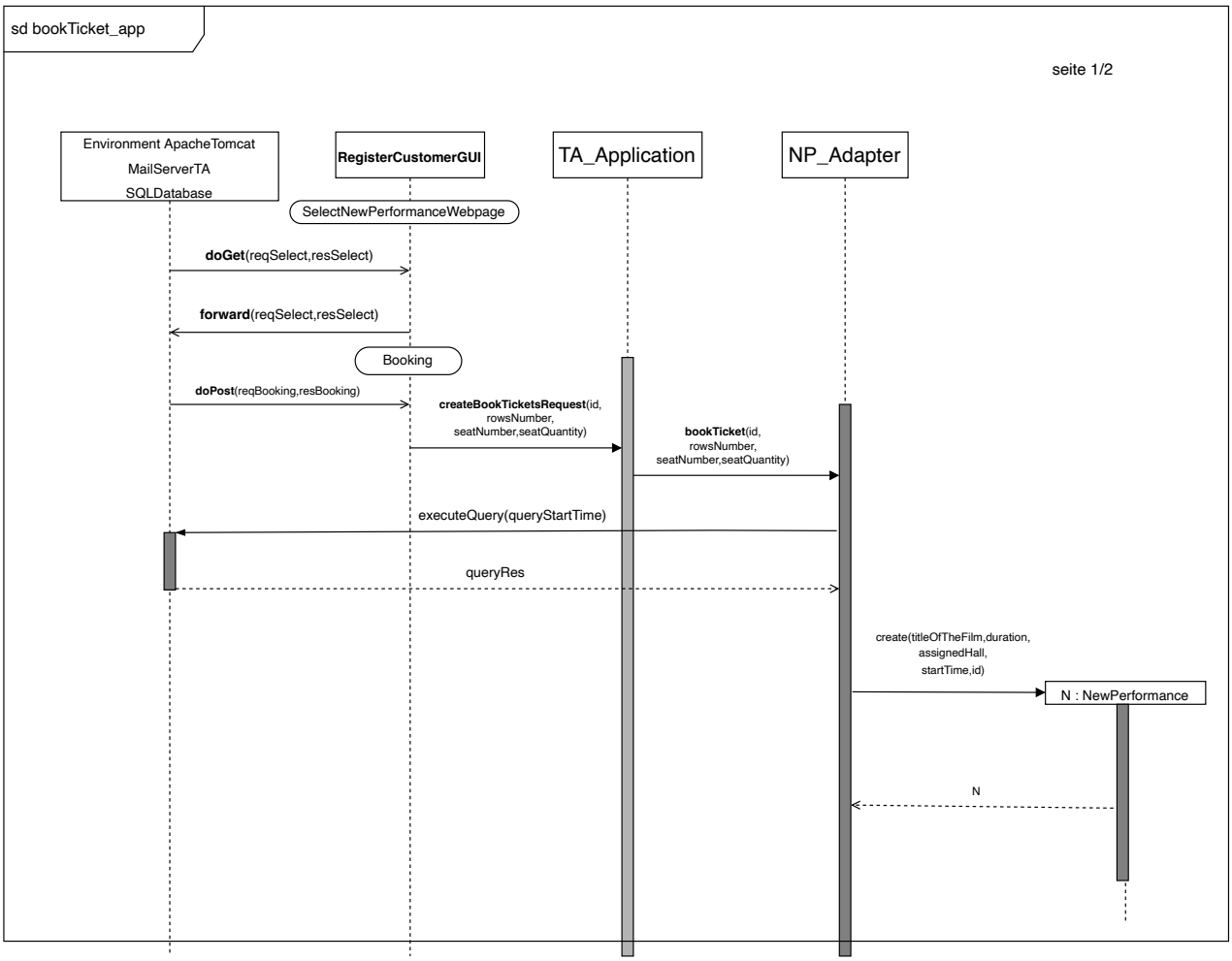
Inter-component Interaction for R2 :

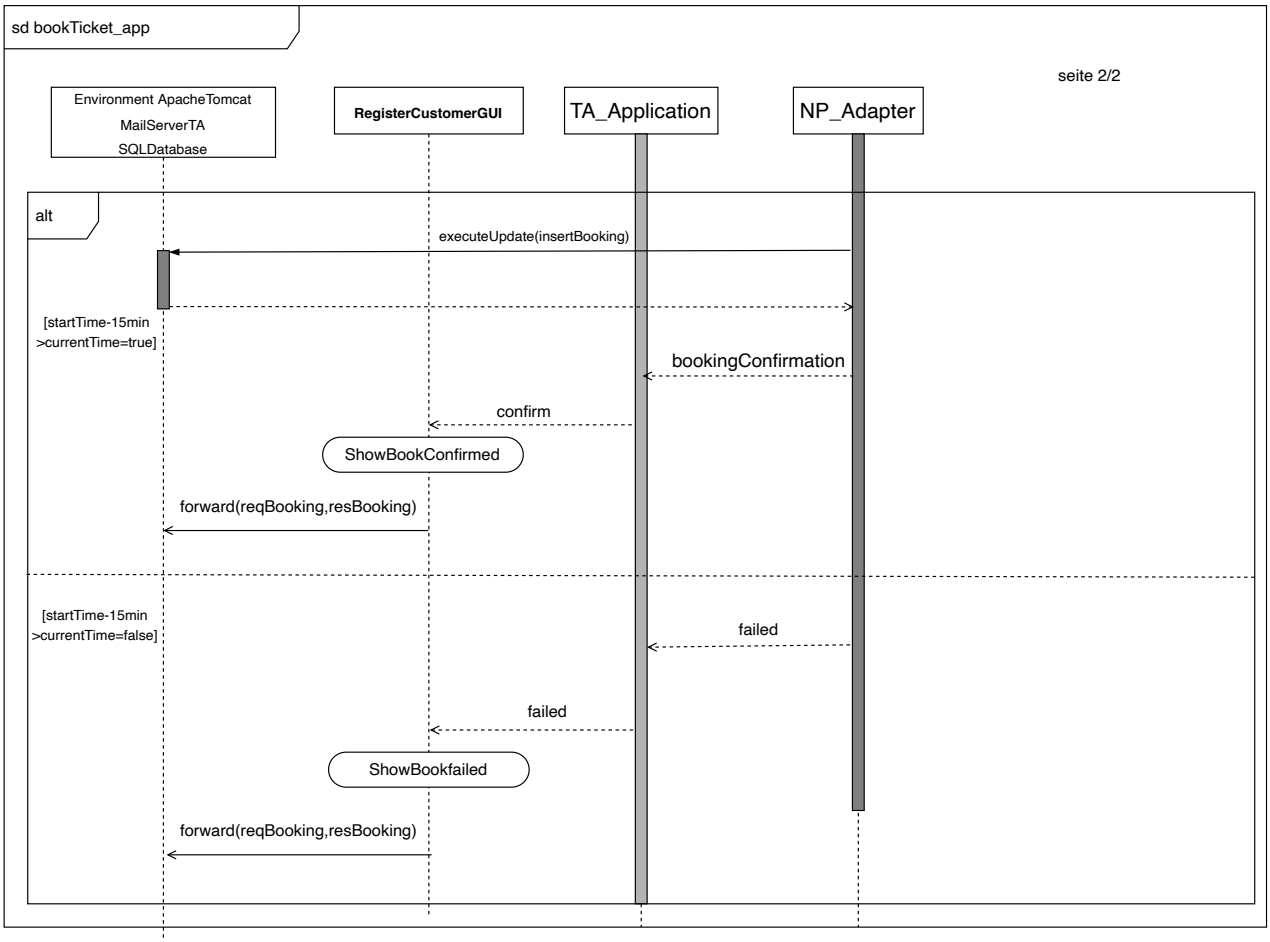


queryPList

```
SELECT * FROM Performance WHERE (startTime>current_Timestamp)
```

Inter-component Interaction for R3 :





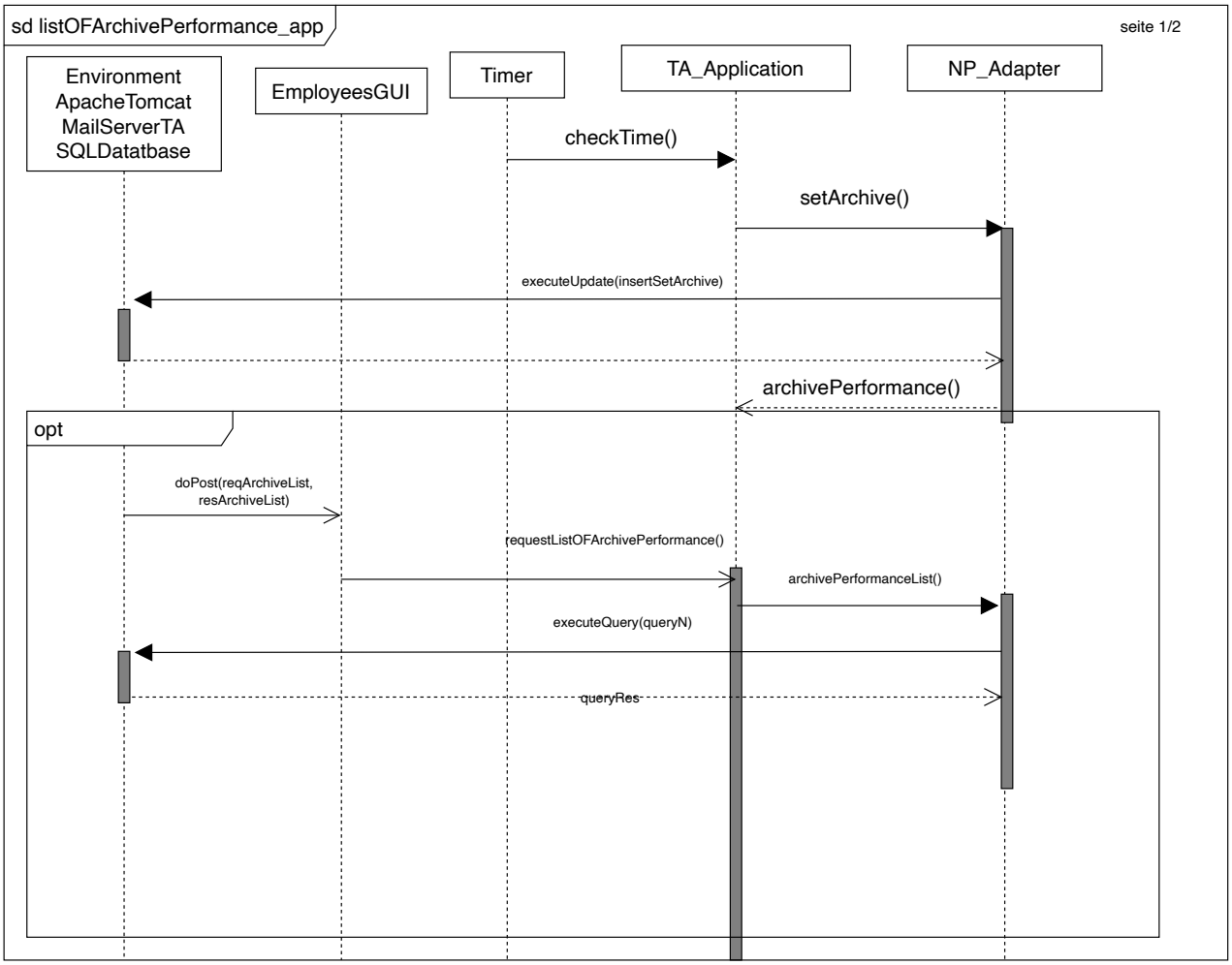
queryStartTime

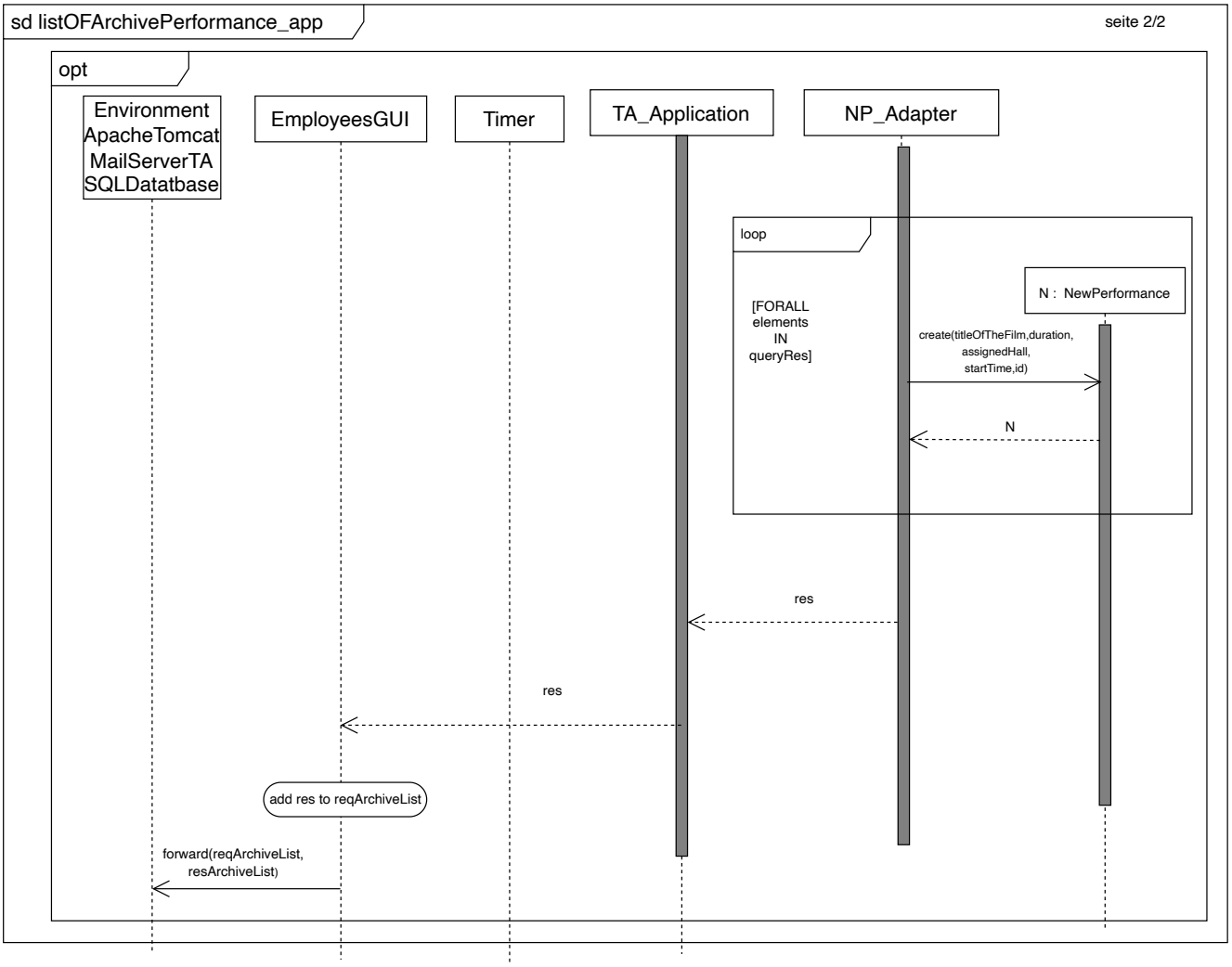
SELECT * FROM Performance WHERE (Timestampadd(startTime,-15min)=> current_Timestamp)

insertBooking

INSERT INTO Booking (id,rowsNumber,seatNumber,seatQuantity) VALUE ("id", "rowsNumber", "seatNumber","seatQuantity")

Inter-component Interaction for R4 :





insertSetArchive

INSERT INTO Archive SELECT * FROM Performance
WHERE (startTime = > current_Timestamp)
DELETE FROM Performance WHERE (startTime = > current_Timestamp)

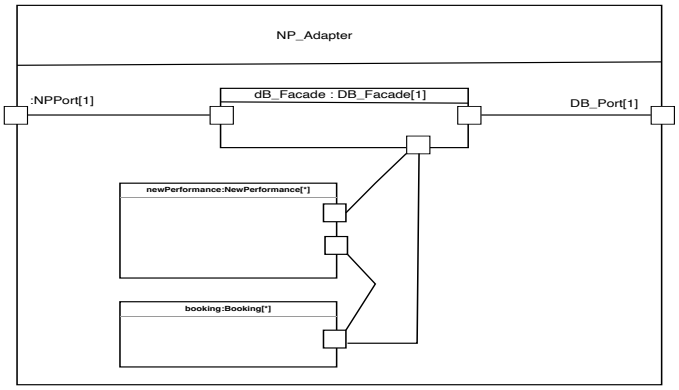
queryN

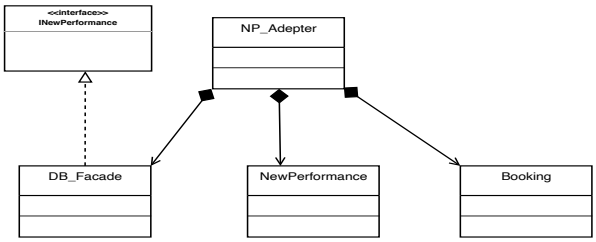
SELECT * FROM Archive

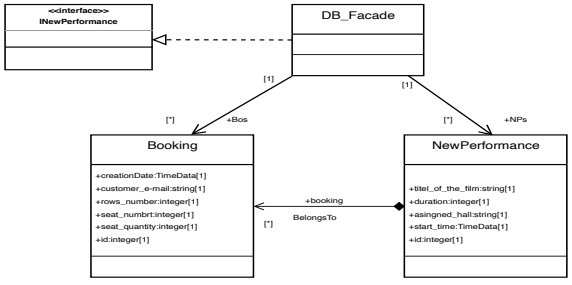
D3: Intra-Component Interaction

Preliminary architectural description of NP_Adapter

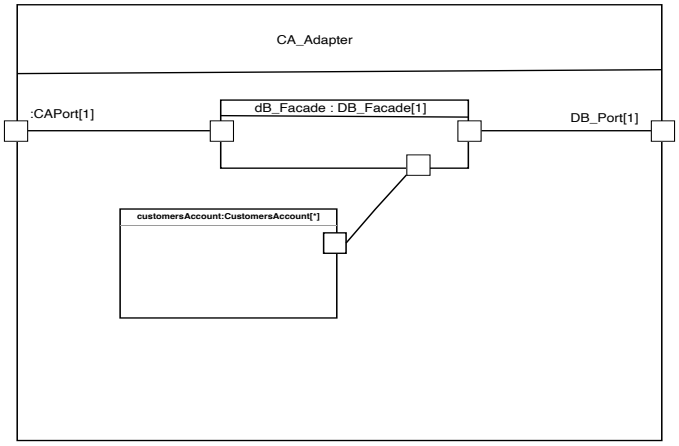
Preliminary architectural description

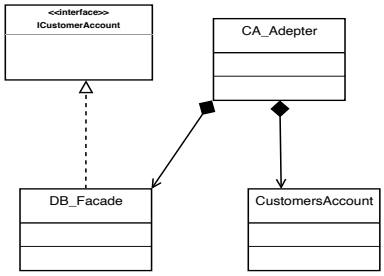


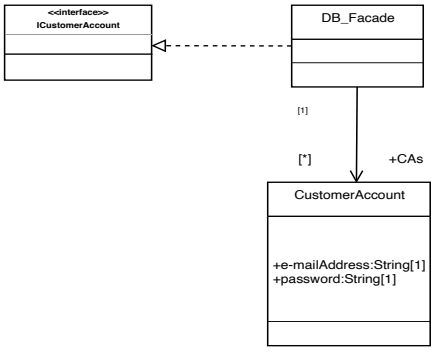




Preliminary architectural description of CA_Adapter







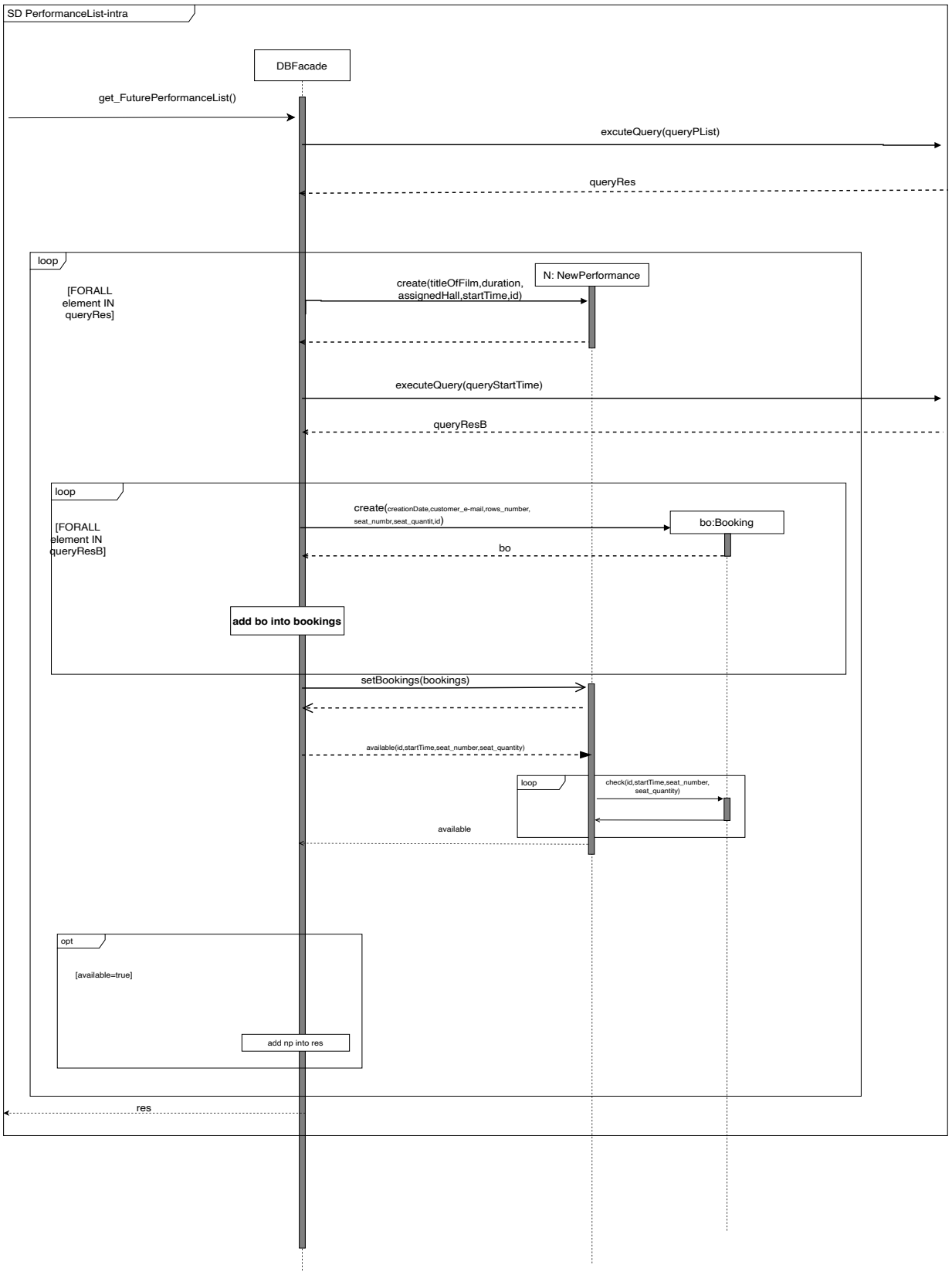
Inter-component interaction for R01

There is no need to specify an intra-component interaction. The CA_Adapter is not complicated and we do not need to split it into sub-components. There is no decomposition of the components defined in Step D1. There is nothing to refine.

Inter-component interaction for R04

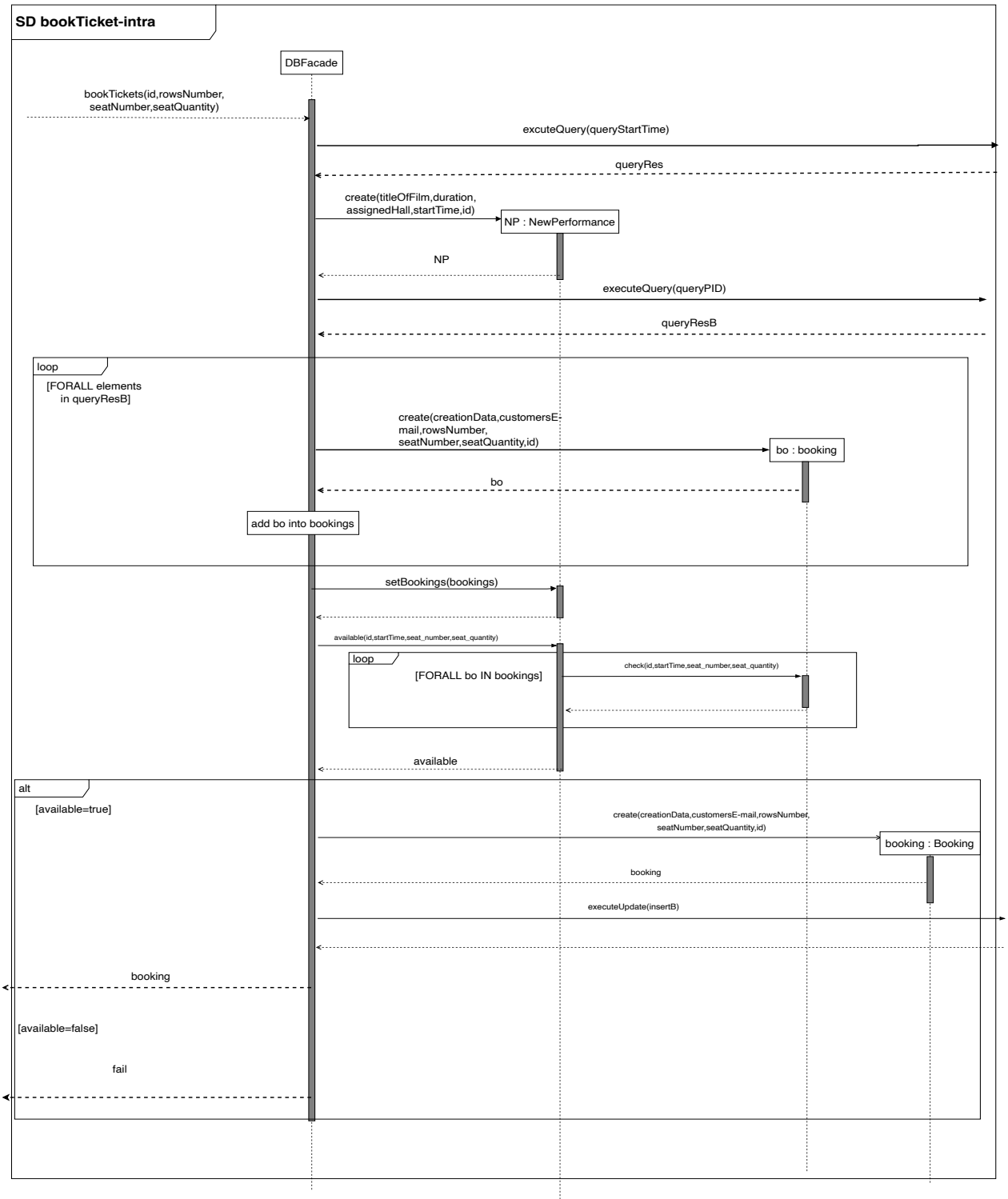
There is no need to specify an intra-component interaction. The NP_Adapter is not complicated and we do not need to split it into sub-components. There is no decomposition of the components defined in Step D1. There is nothing to refine.

Inter-component interaction for R02



queryPList
SELECT * FROM NewPerformance
queryStratTime
SELECT * FROM NewPerformance WHERE(TIMESTAMPADD(SQL_TSI-MINUTE,-15,STARTTIME)<CURRENT_TIMESTAMP)

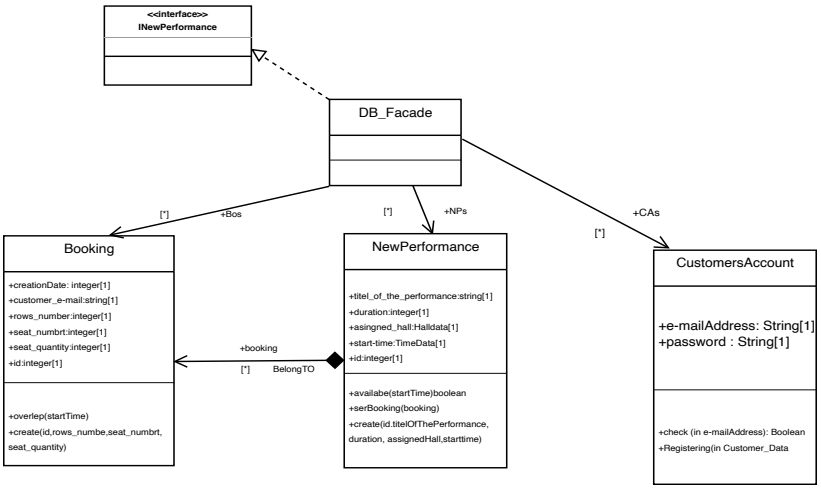
Inter-component interaction for R03



queryStratTime
SELECT * FROM NewPerformance WHERE(TIMESTAMPADD(SQL_TSI-MINUTE,-15,STARTTIME)<CURRENT_TIMESTAMP)
queryPID
Select * from Booking where(np_id = “np.id”)
insertB
INSERT INTO BOOKING (creationData, customersE-mail,rowsNumber,seatNumber,seatQuantity,id)VALUES (“creationData”,”customersE-mail”,”rowsNumber”,”seatNumber”,”seatQuantity”,”id”)

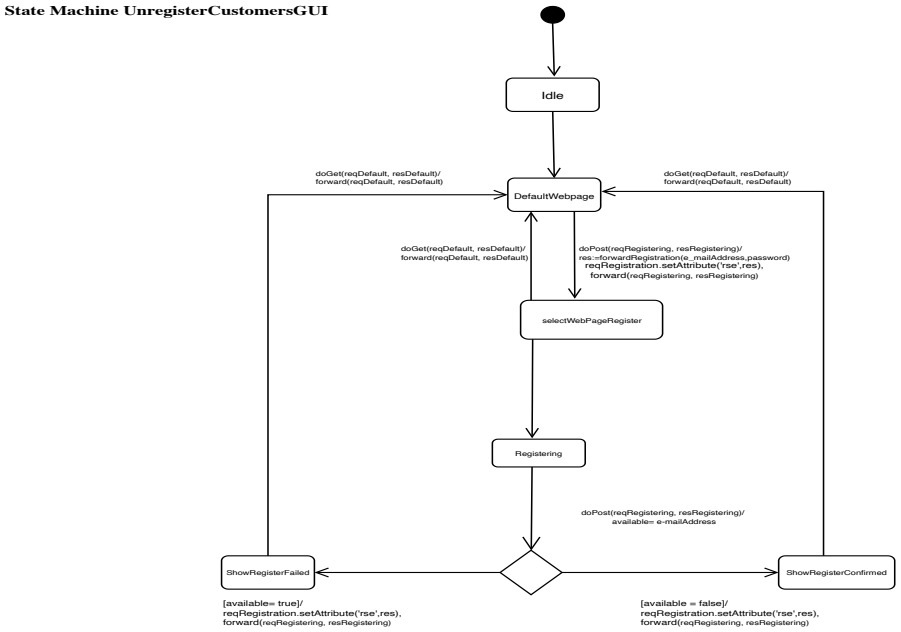
Final architectural description of CA_ NP_Adapter

Final architectural description

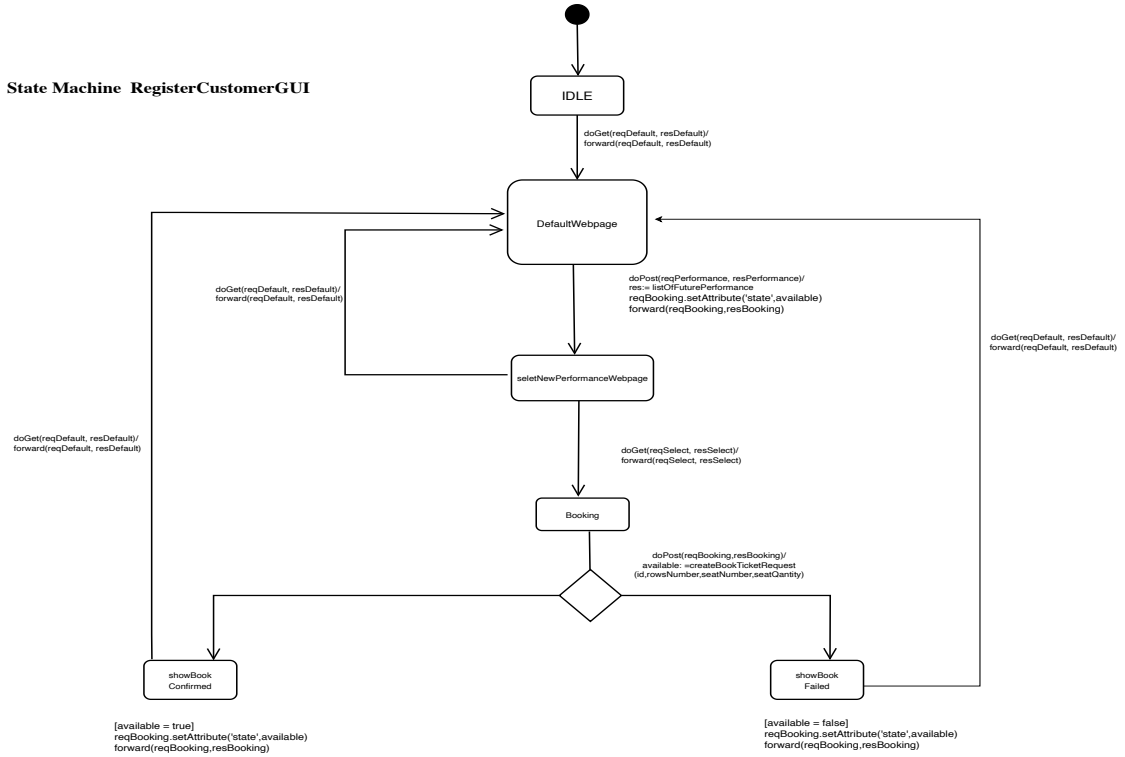


D4: Complete component or class behavior

State Machine UnregisterCustomerGUI



State Machine RegisterCustomerGUI



Glossary

Name	Kind	Discription	Source
A			
archivePerformance	Phenomenon	Counterpart to showArchivedPerformance	CD
ApacheTomcat	connectionDomain	An Open Source JSP and Servlet Container from the Apache Foundation	technicalcontextDigram
B			
bookingFailed	Message	Machine informs Web Page about the booking failure with this	SD_bookTicket
bookingFailedInfo	Message	Web Page informs registered customers about the booking failure with this	SD_bookTicket
bookable	State Predicate	This state confirms that the booking procedure is possible	SD_bookTicket
¬bookable	State Predicate	This state confirms that the booking procedure is not possible	SD_bookTicket
bookTicket	Phenomenon	Booking for performance	CD,pdbookTicket
bookingConfirmation	Phenomenon	Give customers confirmation for the bookings	CD, pdbookTicket
bookTicketsForPerformance	Phenomenon	Can book tickets for a performance	CD, pdbookTicket
bookTicket	class	Represents a performance ticket reserved or booked by a customers	Class model

Booking	state	Indicates the booking procedure	Sd_bookTicket_app
C			
CAAdapterPort			
cacellationNotification	Phenomenon	Notify customers about cacelled performance	CD
confirmRegistration	phenomenon	Machine confirm Unregistered customer about the registration process	pdRegister
customersData	phenomenon	Provide customers data to Machine	CD, pdRegister
Customers_Data	DataType	data of a customer summarized, used as parameters	cdRegister
Customer	class	Representation a customer from customersAccount	cdRegister
confirmRegisterRepresentation	phenomenon	ConnectionDomain representation Unregistered customer about the registration process	pdRegister
cancellationOfPerformances	Phenomenon	Can cancel any performances	CD
creationOfNewPerformancs	Phenomenon	Cancreatenewperformances	CD
CustomersAccount	Domain	Customers account	CD
customersData	Phenomenon	Provides customer data	CD,pdRegister
createBookTickersRequest	Phenomenon	Connection domain create a request to book the ticket to Machine	pdbookTicket
confirmTickerBooking	Phenomenon	Machine confirm the connection domain about the ticket booking process	pdbookTicket
confirmationRepresentation	Phenomenon	Connection domain confirm customers about the ticket booking process	pdbookTicket

creationDate	attribute	Represents the date the booking is created	Class model
Confirm	phenomenon	confirm customers about the ticket booking process, and confirm customers about the registration process	Sd_bookTicket_app, Sd_Register_app
CU_Adapter	component	responsible to create and maintain tables for all persistent classes	subArchRegister globalArch
currentDate()	auxiliary function	Provides the current Date	Class model
CAPort	port	Connects ICustomersAccount to CA_Adapter	subArchRegister globalArch
CAPort_I	port	Connects ICustomerAccount to Application	subArchRegister globalArch
CAAdapterPort	port	Connects the CAAAdapterto to the machine	subArchRegister globalArch
D			
DBFacade	component		Sd_PerformanceList_intra, Sd_bookTicket_intra
Database	Domain	Database	CD
deletePerformancesOlderThan1Year	Phenomenon	Automaticallydeleteperformances older than 1 year	CD
idle	state	ndicates the idle status of web server without any re- quests	State Machine of RegisterCustomerGUI
detailsOfPerformance	Message	Send specific performance details to Machine as reply to the Machine's command	SD_bookTicket

doGet	technical phenomenon	Operations doGet is defined in abstract class javax.servlet.http.HttpServlet (https://docs.oracle.com/javaee/7/api/j avax/servlet/http/HttpServlet.html)	TCD,Sd_register_app, Sd_PerformanceList_app, Sd_bookTicket_app,Sd_listOFArchiveP erformance,StateMachine UnregisteredCustomerGUI, StateMachine RegisteredCustomerGUI
doPost	technical phenomenon	Operations doPost is defined in abstract class javax.servlet.http.HttpServlet (https://docs.oracle.com/javaee/7/api/j avax/servlet/http/HttpServlet.html)	TCD,Sd_register_app, Sd_PerformanceList_app,Sd_bookTick et_app,Sd_listOFArchivePerformance, StateMachine UnregisteredCustomerGUI, StateMachine RegisteredCustomerGUI
DefaultWebpage	state	Indicates the starting page	State Machine of RegisterCustomerGUI , State Machine of UnregisterCustomerGUI
E			
e-mailAddressNotUnique	Message	Web Page send this message to Unregistered Customers as the emails they put were also used in previous times	SD_Register
Employees	Domain	Employees	CD
EmployeesGUI	component	web interface for Employees	subArchListOFArchivePerformance globalArch
ECmdsPort	port	connects ECmds to the application	subArchListOFArchivePerformance globalArch
ECmdsPort	port	connects RCCmds to the application	subArchListOFArchivePerformance globalArch
executeQuery	message	Java API function to send an SQL query command to a MySQL database.	TCD, Sd_register_app Sd_PerformanceList_app Sd_bookTicket_app Sd_listOFArchivePerformance

executeUpdate ()	message	Java API function to send an SQL query command to a MySQL database.	TCD, Sd_register_app Sd_PerformanceList_app Sd_bookTicket_app Sd_listOFArchivePerformance
ECmdsPort_I	port	Connects to the EmployeesGUI	subArchListOFArchivePerformance globalArch
F			
forwardRegistration	Phenomenon	Connection domain forward the registration request to machine for unregistered customer	pdRegister
forwardPerformanceList	Phenomenon	Lexical domain Forwawrd performance list to machine	pdperformanceList
FeedbackUC	Phenomenon	Necessary feature to notify UnregisteredCustomers wether their request was done successfully or not Counterpart to register	CD
FeedbackRC	Phenomenon	Necessary feature to notify RegisteredCustomers wether their request was done successfully or not; Counterpart to login, logout, bookTicketsForPerformancs, viewingFuturePerformances	CD
feedbackOfNewPerformances	Phenomenon	Machine give confirmation to employees that new performance has been set	CD
feedbackOfArchivedPerformance	Phenomenon	Machine present archived performances to employees	CD
feedbackOfCancellationOfPerformances	Phenomenon	Machine confirm to employees about Cancellation Of Performances	CD
failedRegistration	Message	Machine confirms Web Page about the registration failure	SD_Register

forward	technical Phenomenon	Operation forward defined in interface javax.servlet.RequestDispatcher (http://docs.oracle.com/javaee/7/api/javax/servlet/RequestDispatcher.html	TCD, Sd_register_app Sd_PerformanceList_app Sd_bookTicket_app Sd_listOFArchivePerformance
failed	Phenomenon	the response from the database, when exists condition is false for SD_Register _app and SD_bookTicket _app.	Sd_register_app Sd_bookTicket_app
G			
get_detailsOfPerformance	Message	To get a specific performance details form the lexical domain(NewPerformance)	SD_bookTicket
get_FuturePerformanceList	Message	Machine gives this to New Performance to get the performance details as needed	SD_PerformanceList
get_Customers_Data	Message	Machine gives this command to CustomersAccount (Lexical Doamin) to get all the registered Customers data (emails).	SD_Register
I			
Id	attribute	represents unique id of new Performance represents unique id of booking	Class model
ITimer	interface	used to trigger the inter- nal operation “checkTime” periodically	subArchBookTicket subArchListOFArchivePerformance globalArch
INewPerformance	interface		
L			
Login	Phenomenon	Have to login to use the functionalities provided by the ticket-booking app	CD

LoggingOut	Phenomenon	Counter part to logout	CD
Logout	Phenomenon	Can also logout	CD
LoggingIn	Phenomenon	Counterpart to login	CD
listOfFuture Performances	Phenomenon	Display the list of performances that are in the future	CD
listOFArchivePerformanceRepresentation	Phenomenon	Connection domain represent the archived performances to the employees	pdlistOFArchivePerformance
N			
newPerformance	Domain	New performance	CD
NP_Adapter	component	responsible to create and maintain tables for all per-sistent classes	subArchPerformanceList subArchBookTicket subArchListOFArchivePerformance globalArch
NPAdapterPort	port	Connects the NPAdapterto to the machine	subArchPerformanceList subArchBookTicket subArchListOFArchivePerformance globalArch
NPPort	port	Connects INewPerformance to NPAdapter	subArchPerformanceList subArchBookTicket subArchListOFArchivePerformance globalArch
NPPort_I	port	Connects INewPerformance to Application	subArchPerformanceList subArchBookTicket subArchListOFArchivePerformance globalArch
P			
provideListOFArchivePerformance	phenomenon	Machine provides the list of Archived Performances to the connection domain	pdlistOFArchivePerformance

performanceListRepresentation	phenomenon	Connection domain represent the performance list to registered customers	pdperformanceList
R			
requestListOFArchivePerformance	phenomenon	Connection domain request to machine for the list of Archived performance	pdlistOFArchivePerformance
RegisterCustomerPort	port	Connects the RegisterCustomerGUI to the machine	subArchPerformanceList subArchbookTicket globalArch
RCCmdsPort	Port	connects RCCmds to the application	subArchPerformanceList subArchbookTicket globalArch
RCCmdsPort_I	Port	Connects to the RegisterCustomerGUI	subArchPerformanceList subArchbookTicket globalArch
RCCmds	interface	Provides Operation and functions for interaction with RegisterCustomerGUI	subArchPerformanceList subArchbookTicket globalArch
RegisteredCustomeGUI	component	web interface for RegisteredCustome	subArchPerformanceList subArchbookTicket globalArch
registering	phenomenon	Machine access to the lexical domain for the registration of a new customer	CD, pdRegister
register	phenomenon	Unregister customer request for registration	CD, pdRegister
registrening	Phenomenon	Register new customer account	CD
RegisteredCustome	Domain	Registered Customers	CD
Register	Phenomenon	Have to register	CD
RegisterCustomerWebBrowser	connectionDomain	Web browser used by RegisterCustomer	technicalcontexDigram

S			
ShowBookConfirmed	state	Indicates that ticket bookinghas been completed	stateMachineregisteredCustomer GUI
ShowBookfailed	state	Indicates that ticket booking has been completed	stateMachineregisteredCustomer GUI
SelectNewPerformanceWebpage	state	Indicates that performance List	stateMachine
ShowRegisterConfirmed	state	Indicates that registration has been completed	StateMachine unregisteredCustomerGUI
ShowRegisterFailed	state	Indicates that registration has been failed	StateMachine unregisteredCustomerGUI
setBookings(. . .)	message	Sets the association between the collection of Booking objects and the NewPerformance	Sd bookticket_intra
showAvailablePerformancesList	phenomenon	Registered customer ask connection domain to show the available performance list.	CD, pdperformanceList
sortedListOfFuturePerformances	phenomenon	Machine provides connection domain the sorted list of future performance for registered customers	pdperformanceList
setNewPerformance	Phenomenon	Counterpart to creationOfNewPerformance	CD
setArchive	Phenomenon	Machine set performances to lexical domain as archived	CD
SQL	Tecnical Phenomenon	defined in FIPS PUB 127-2	technicalcontextDiagram

showAvailablePerformancesList	Phenomenon	Can view the future performances	CD
showArchivedPerformance	Phenomenon	Display the list of performances that are in Archived	CD,pd ListOFArchivePerforma nce
storeShowedPerformance	Phenomenon	Store performances which are already showed	CD
showListOfFuturePerformances	Phenomenon	Connection domain ask machine to show the list of future performances	pdperformanceList
T			
TA-bookTicket	Domain	Part of the machine	pdbookTicket
TA_Register	Domain	Part of the machine	pdRegister
Ticket-Booking App	Domain	The software we are going to build	CD
TA_ArchiveList	Domain	Part of the machine	pdListOFArchivePerfor
TimerPort	port	Connects ITimer to Timer	subArchBookTicket subArchListOFArchivePerfo rmance globalArch
TimerPort_I	port	Connects to the reusedComponent Timer	subArchBookTicket subArchListOFArchivePerfo rmance globalArch
TA_List	Domain	Part of the machine	pdperformanceList
Timer	reused component	given component initiat- ing the internal operation “checkTime”	subArchBookTicket subArchListOFArchivePerfo rmance globalArch
TimeData	DataType	Data type that represents an exact point in time	cdTA_List
U			
UnregisterCustomer	Domain	Unregistered Customers	CD
uniqueE-mail	State Prediacte	It stands for the registration process is possible	SD_Register
¬uniqueE-mail	State Predicate	It stands fort he registration process is not possible	SD_Register

UnregisterCustomerGUI	component	web interface for UnregisterCustomer	subArchRegister globalArch
UCCmdsPort	port	connects RCCmds to the application	subArchRegister globalArch
UCCmdsPort_I	port	Connects to the UnregisterCustomerGUI	subArchRegister globalArch
UCCmds	interface	Provides Operation and functions for interaction with unregisterCustomerGUI	subArchRegister globalArch
UnregisteredCustomersPOrt	port	Connects the UnregisterCustomerGUI to the machine	subArchRegister globalArch
W			
WebPageRegister	Domain	The connection domain between the unregister customer and the machine. Forwarding the inputs of the unregister customer to the machine and the outputs of the machine to the unregister customer.	pdRegister,
WebpageOfFuturePerformances	Domain	The connection domain between the Register customer and the machine. Forwarding the inputs of the Register customer to the machine and the outputs of the machine to the Register customer.	pdperformanceList
WebpageOfArchivedList	Domain	The connection domain between the Employees and the machine. Forwarding the inputs of the Employees to the machine and the outputs of the machine to the Employees.	PdListOFArchivePerformance
WebPage-bookTicket	Domain	The connection domain between the Registered customer and the machine. Forwarding the inputs of the Registered customer to the machine and the outputs of the machine to the Registered customer.	pdbookTicket