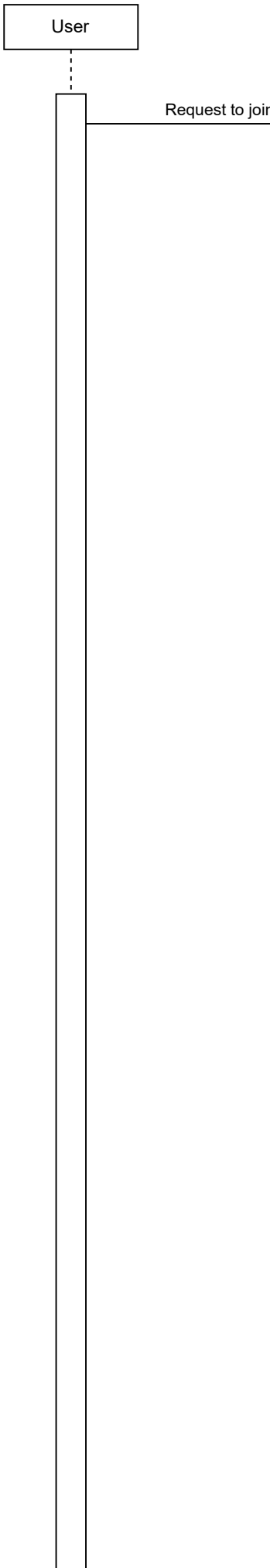
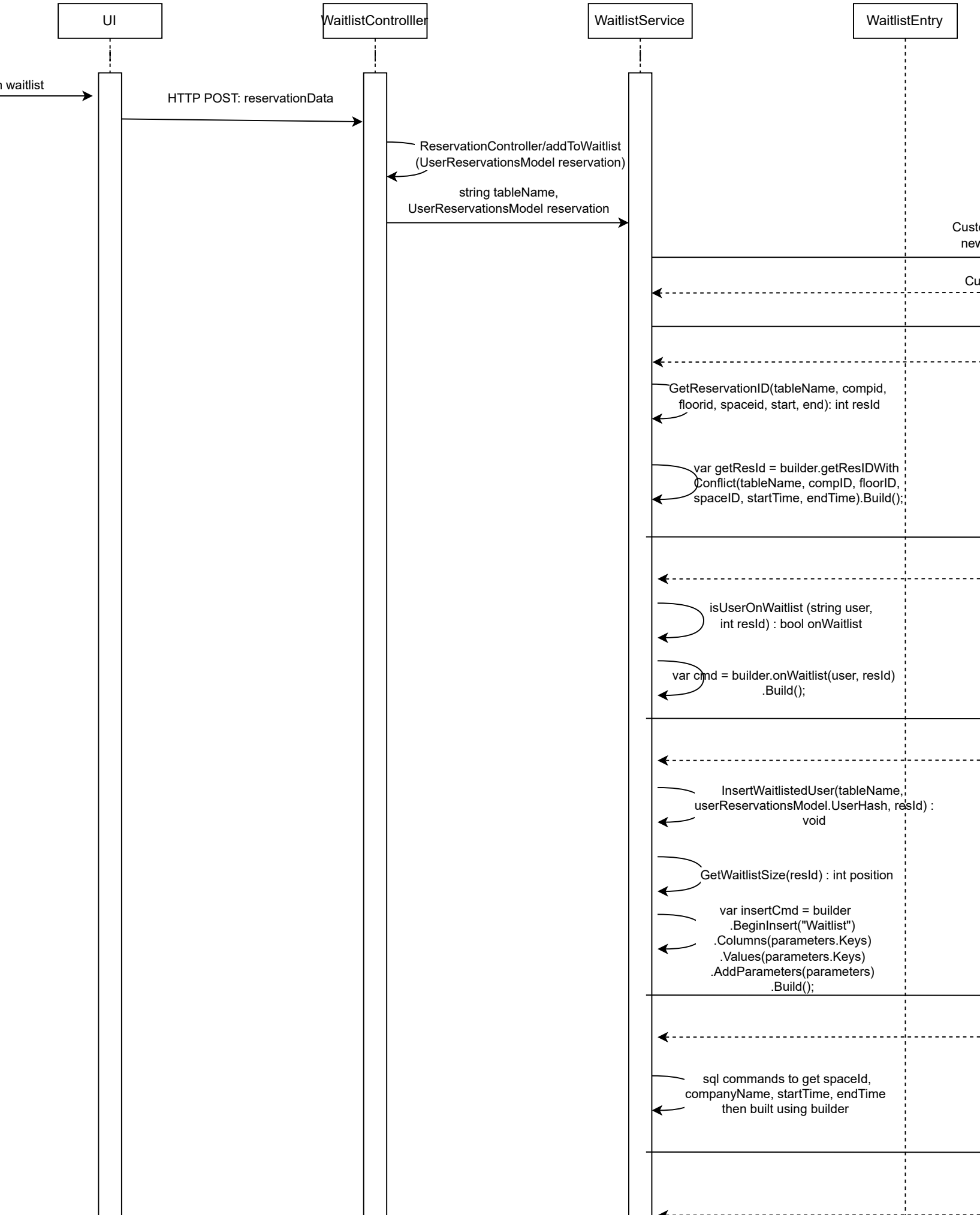
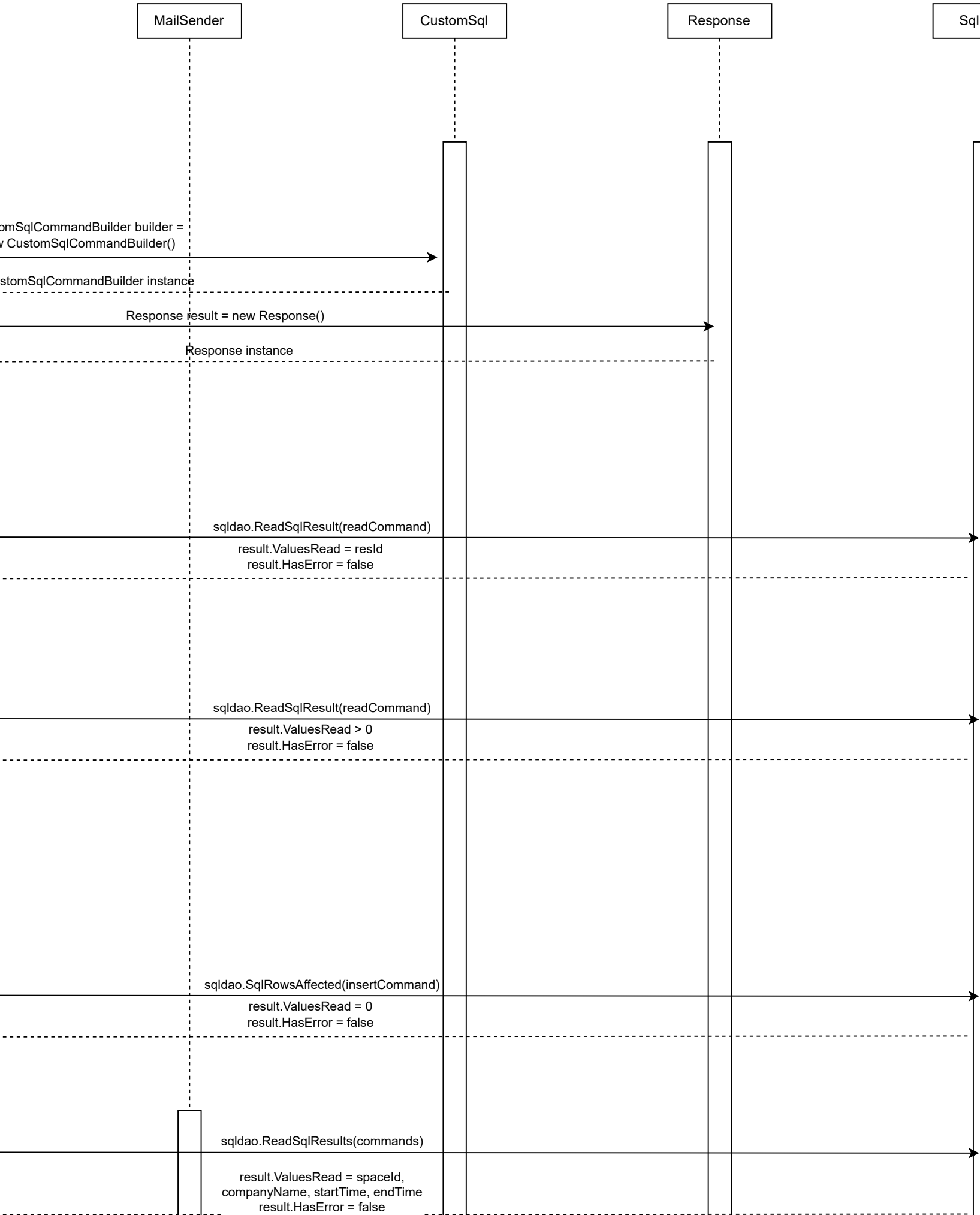
Join Waitlist - Suc

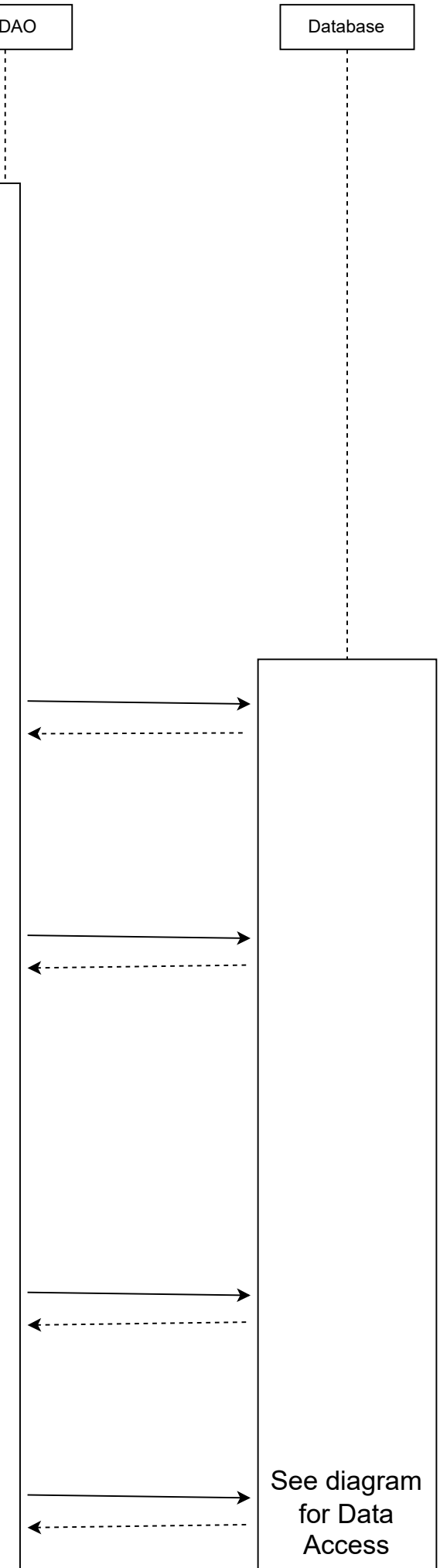
ccess

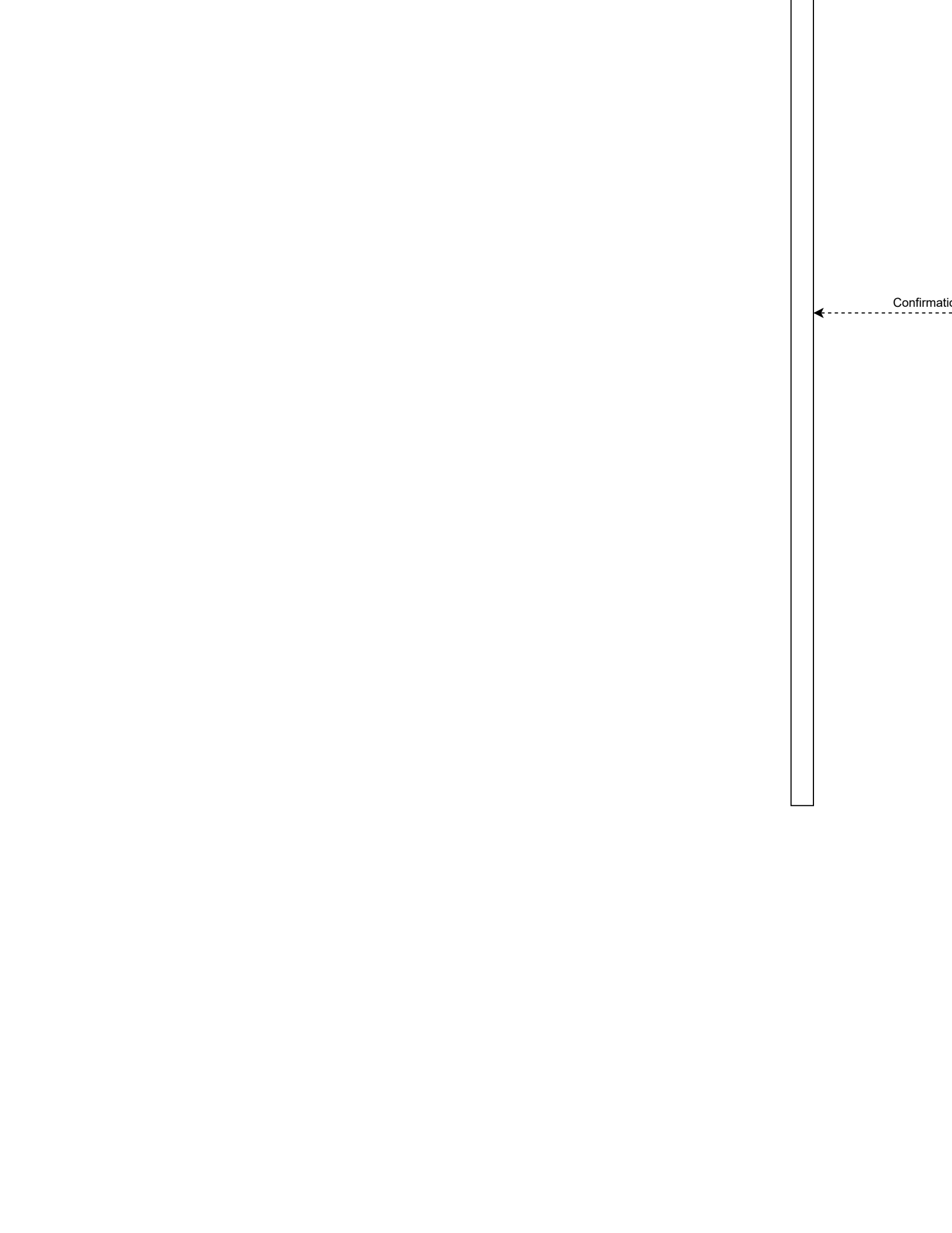
```
reservationData = {
    companyId: int companyId
    floorPlanId: int floorPlanId
    spaceId: string spaceId'
    reservationStartTime: datetime reservation-startTime
    reservationEndTime: datetime reservation-endTime')
    userHash: string username
}
```

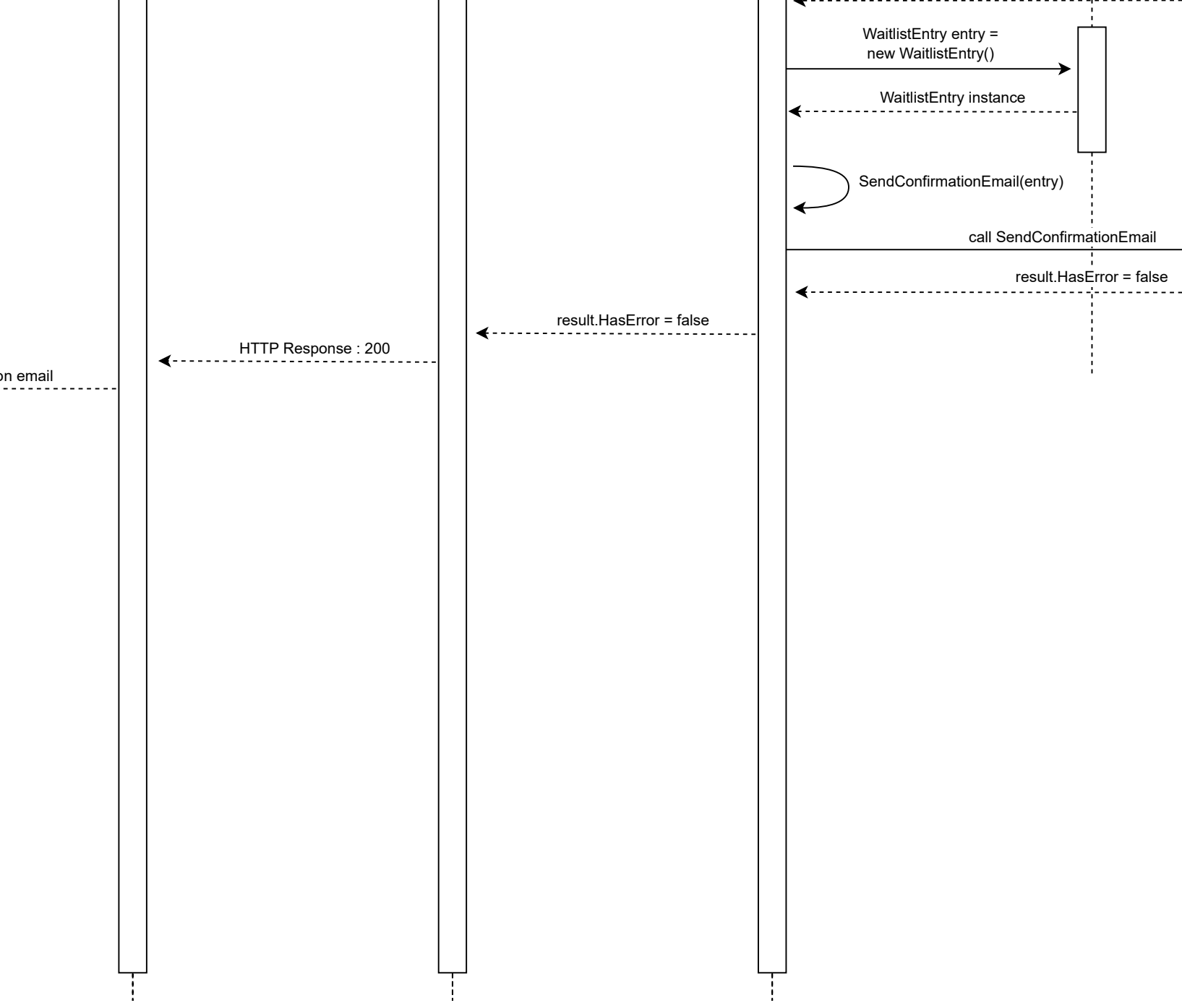


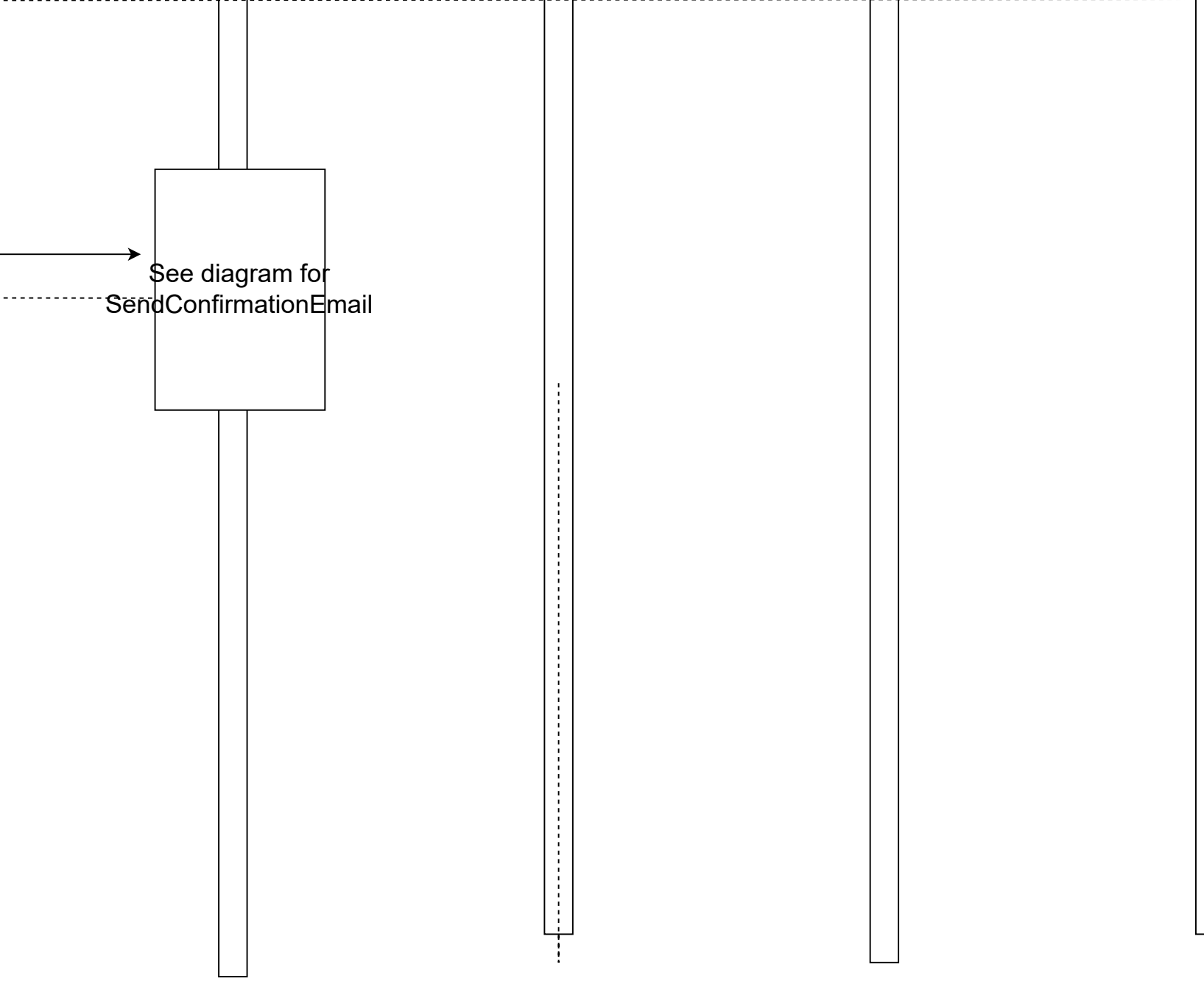

UI
WaitlistControlller
WaitlistService
WaitlistEntry
n waitlist
HTTP POST: reservationData
ReservationController/addToWaitlist
(UserReservationsModel reservation)
string tableName,
UserReservationsModel reservation
Custo
new
Cu
GetReservationID(tableName, compid,
floorid, spaceid, start, end): int resId
var getResId = builder.getResIDWith
Conflict(tableName, compID, floorID,
spaceID, startTime, endTime).Build();
isUserOnWaitlist (string user,
int resId) : bool onWaitlist
var cmd = builder.onWaitlist(user, resId)
.Build();
InsertWaitlistedUser(tableName,
userReservationsModel.UserHash, resId) :
void
GetWaitlistSize(resId) : int position
var insertCmd = builder
.BeginInsert("Waitlist")
.Columns(parameters.Keys)
.Values(parameters.Keys)
.AddParameters(parameters)
.Build();
sql commands to get spaceId,
companyName, startTime, endTime
then built using builder

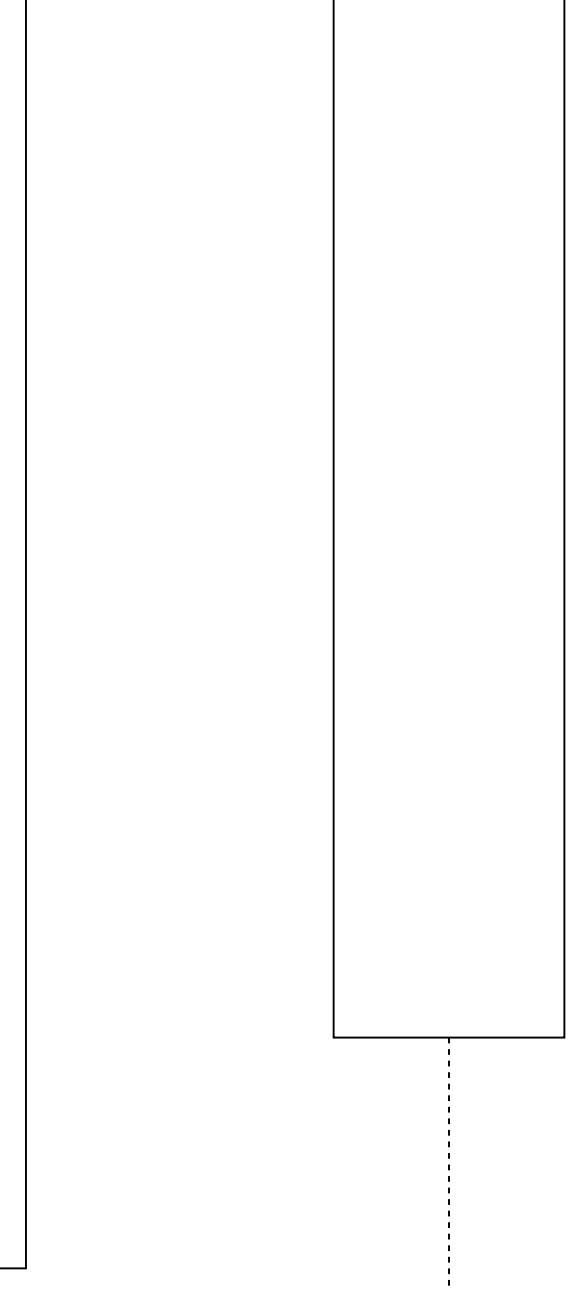

MailSender
CustomSql
Response
Sql
omSqlCommandBuilder builder =
w CustomSqlCommandBuilder()
stomSqlCommandBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResult(readCommand)
result.ValuesRead = resId
result.HasError = false
sqldao.ReadSqlResult(readCommand)
result.ValuesRead > 0
result.HasError = false
sqldao.SqlRowsAffected(insertCommand)
result.ValuesRead = 0
result.HasError = false
sqldao.ReadSqlResults(commands)
result.ValuesRead = spaceId,
companyName, startTime, endTime
result.HasError = false


DAO
Database
See diagram for Data Access


Confirmatio


WaitlistEntry entry =
new WaitlistEntry()
WaitlistEntry instance
SendConfirmationEmail(entry)
call SendConfirmationEmail
result.HasError = false
result.HasError = false
HTTP Response : 200
on email

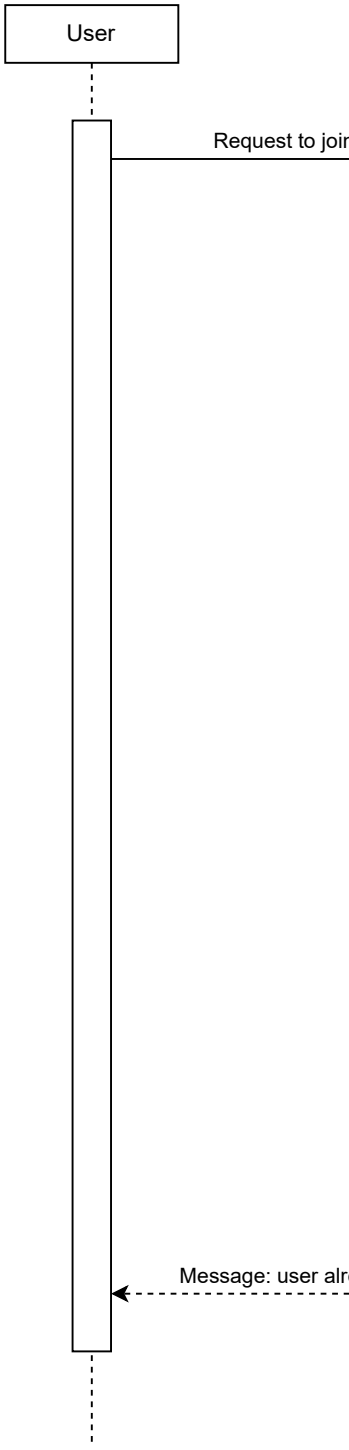

See diagram for
SendConfirmationEmail

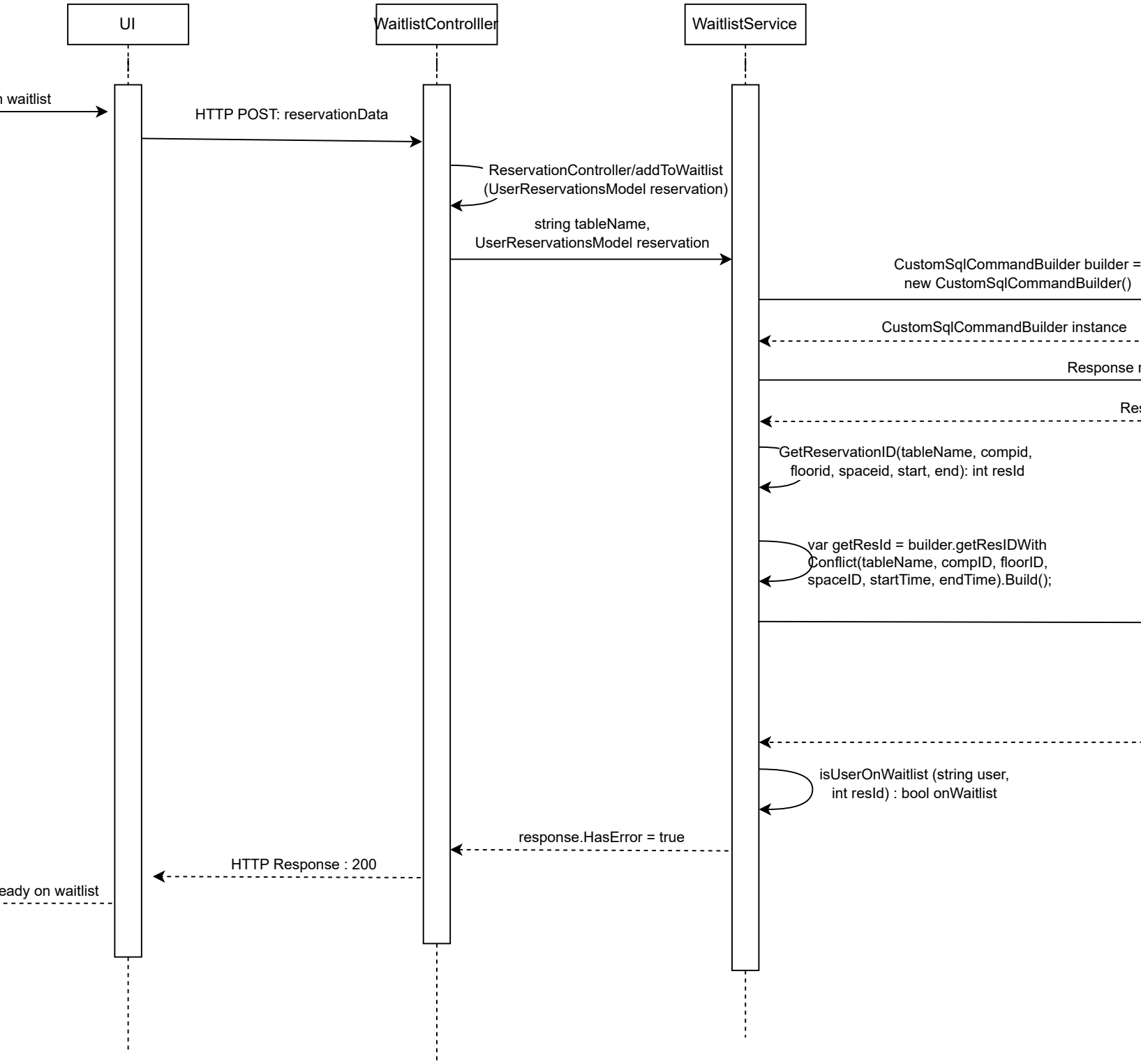


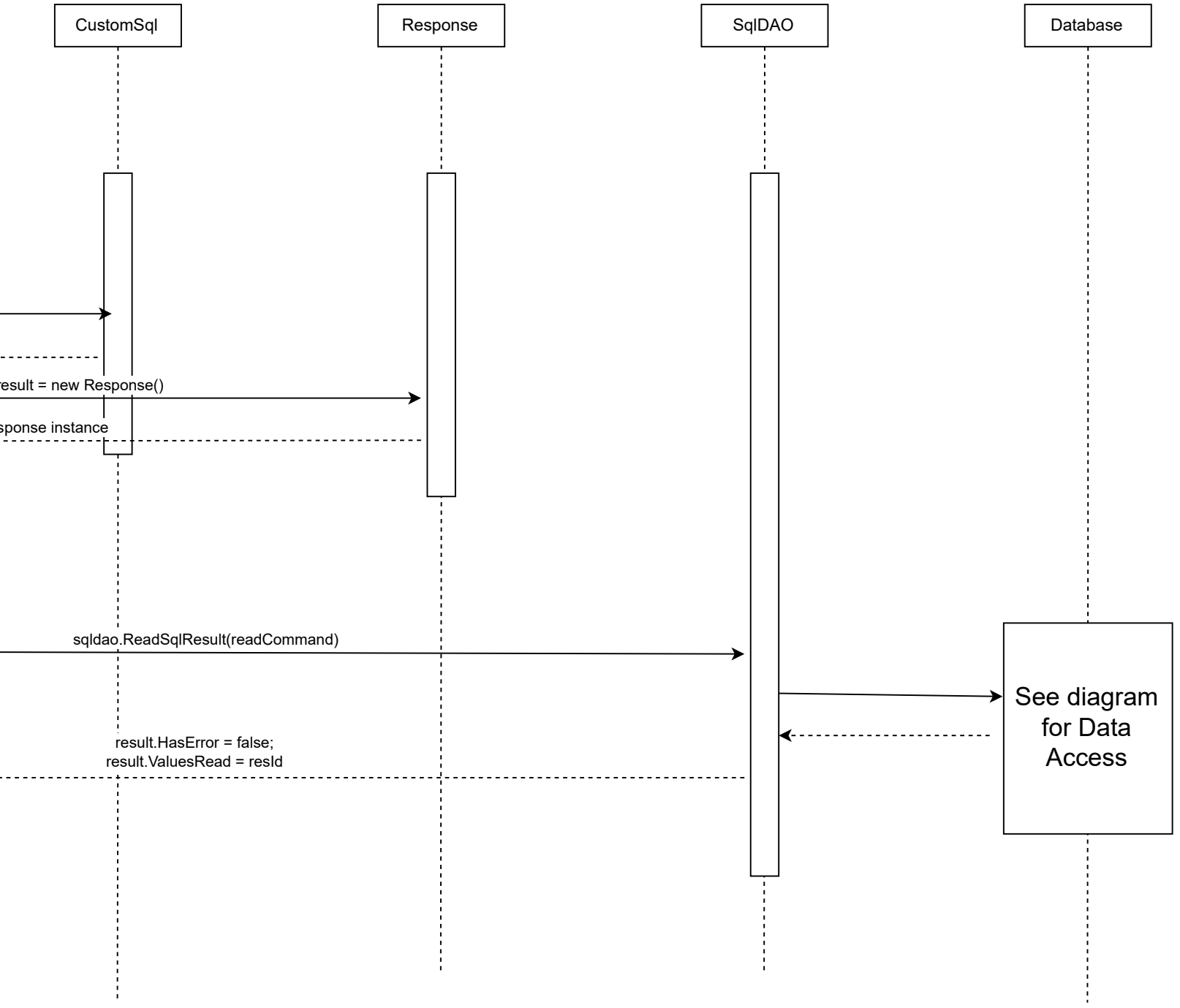
Join Waitlist - Fa

ilure

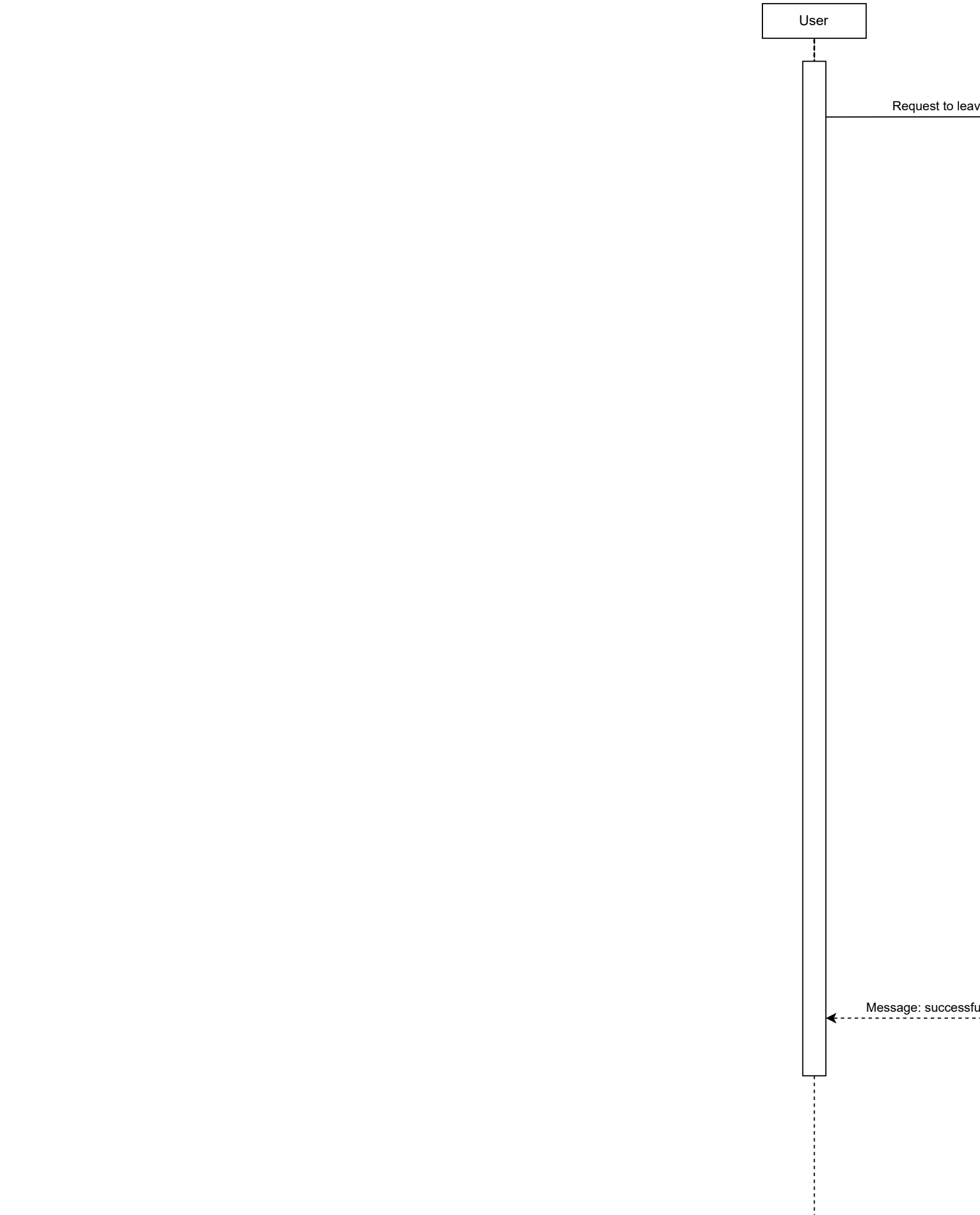
```
reservationData = {
    companyId: int companyId
    floorPlanId: int floorPlanId
    spaceId: string spaceId'
    reservationStartTime: datetime reservation-startTime
    reservationEndTime: datetime reservation-endTime')
    userHash: string username
}
```

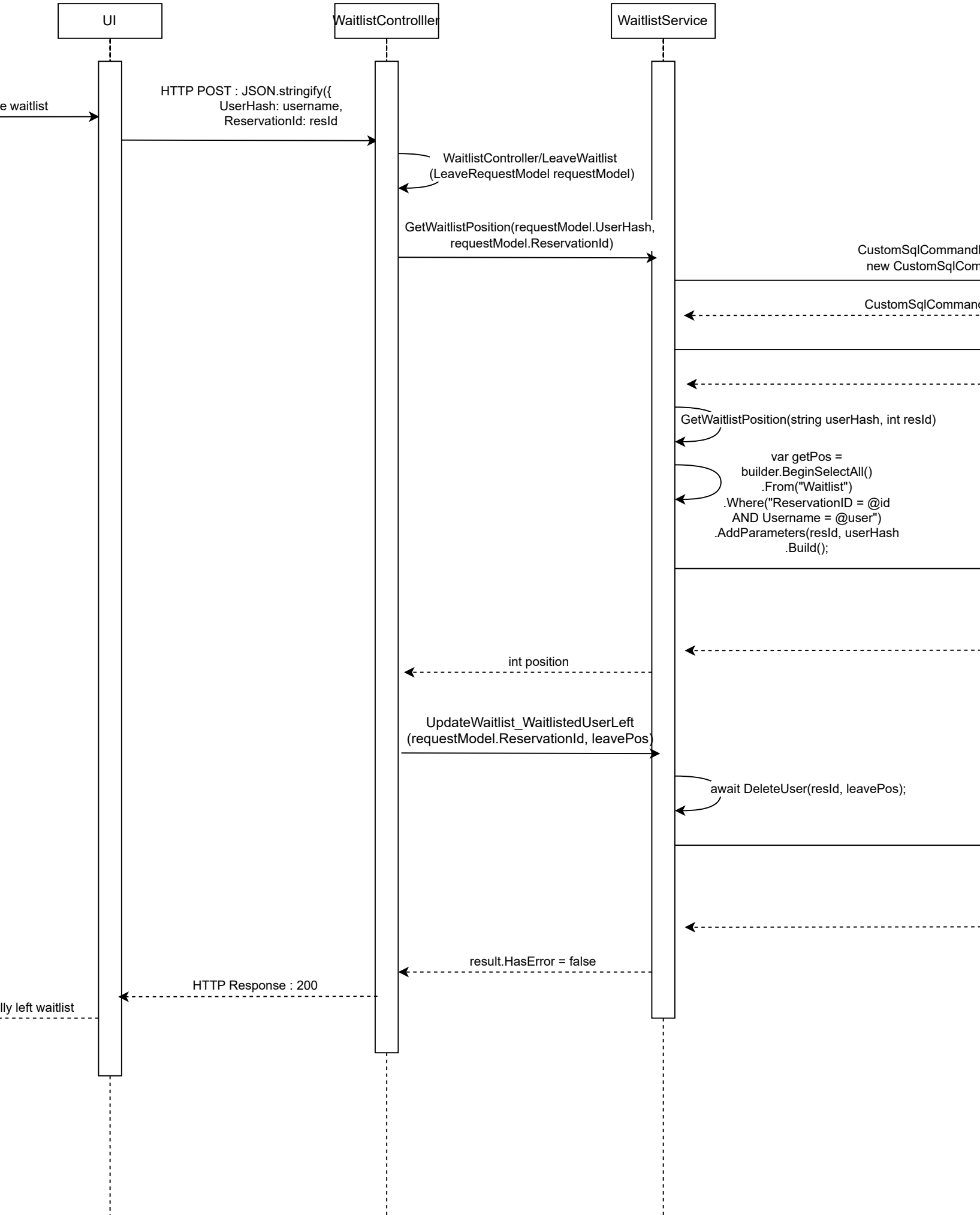


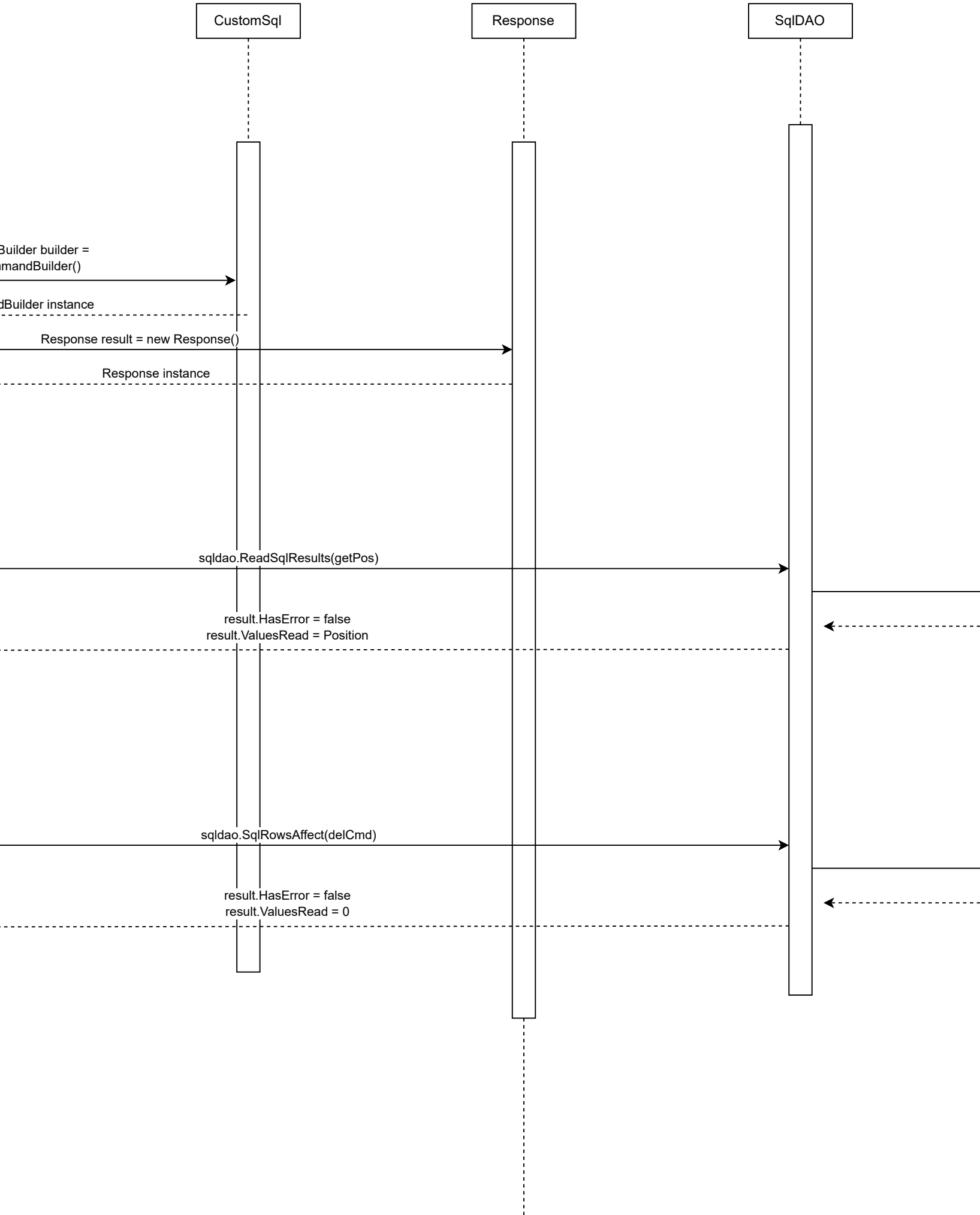

UI
WaitlistControlller
WaitlistService
n waitlist
HTTP POST: reservationData
ReservationController/addToWaitlist
(UserReservationsModel reservation)
string tableName,
UserReservationsModel reservation
CustomSqlCommandBuilder builder =
new CustomSqlCommandBuilder()
CustomSqlCommandBuilder instance
Response
Res
GetReservationID(tableName, compid,
floorid, spaceid, start, end): int resId
var getResId = builder.getResIDWith
Conflict(tableName, compID, floorID,
spaceID, startTime, endTime).Build();
isUserOnWaitlist (string user,
int resId) : bool onWaitlist
response.HasError = true
HTTP Response : 200
eady on waitlist

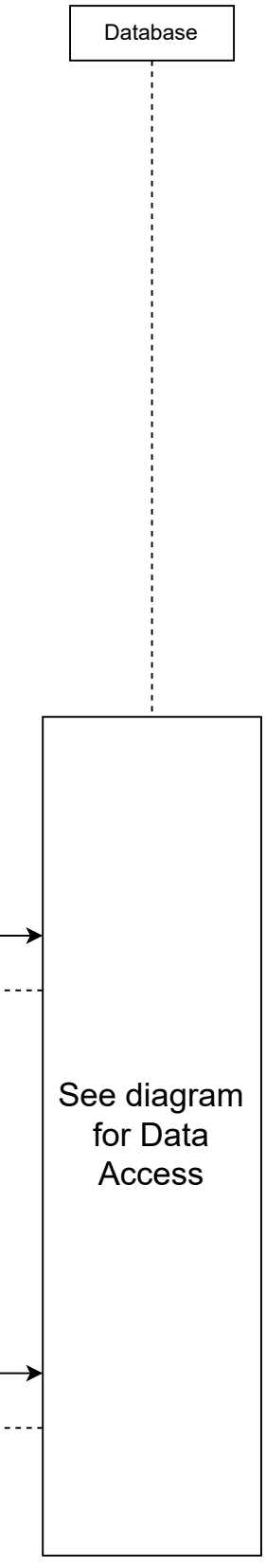

CustomSql
Response
SqlDAO
Database
result = new Response()
sponse instance
sqldao.ReadSqlResult(readCommand)
See diagram for Data Access
result.HasError = false;
result.ValuesRead = resId

Leave Waitlist

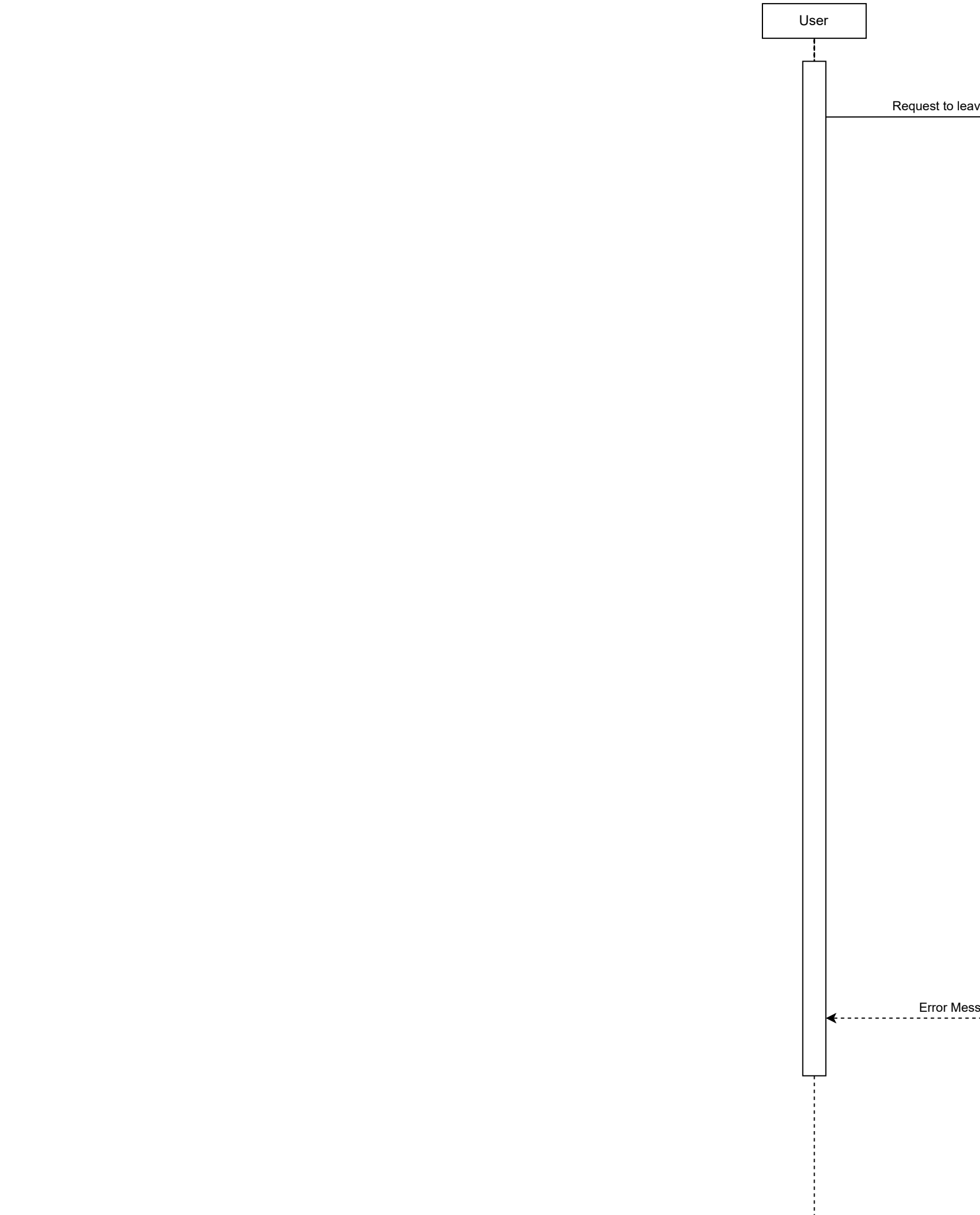

User
Request to leav
Message: successfu

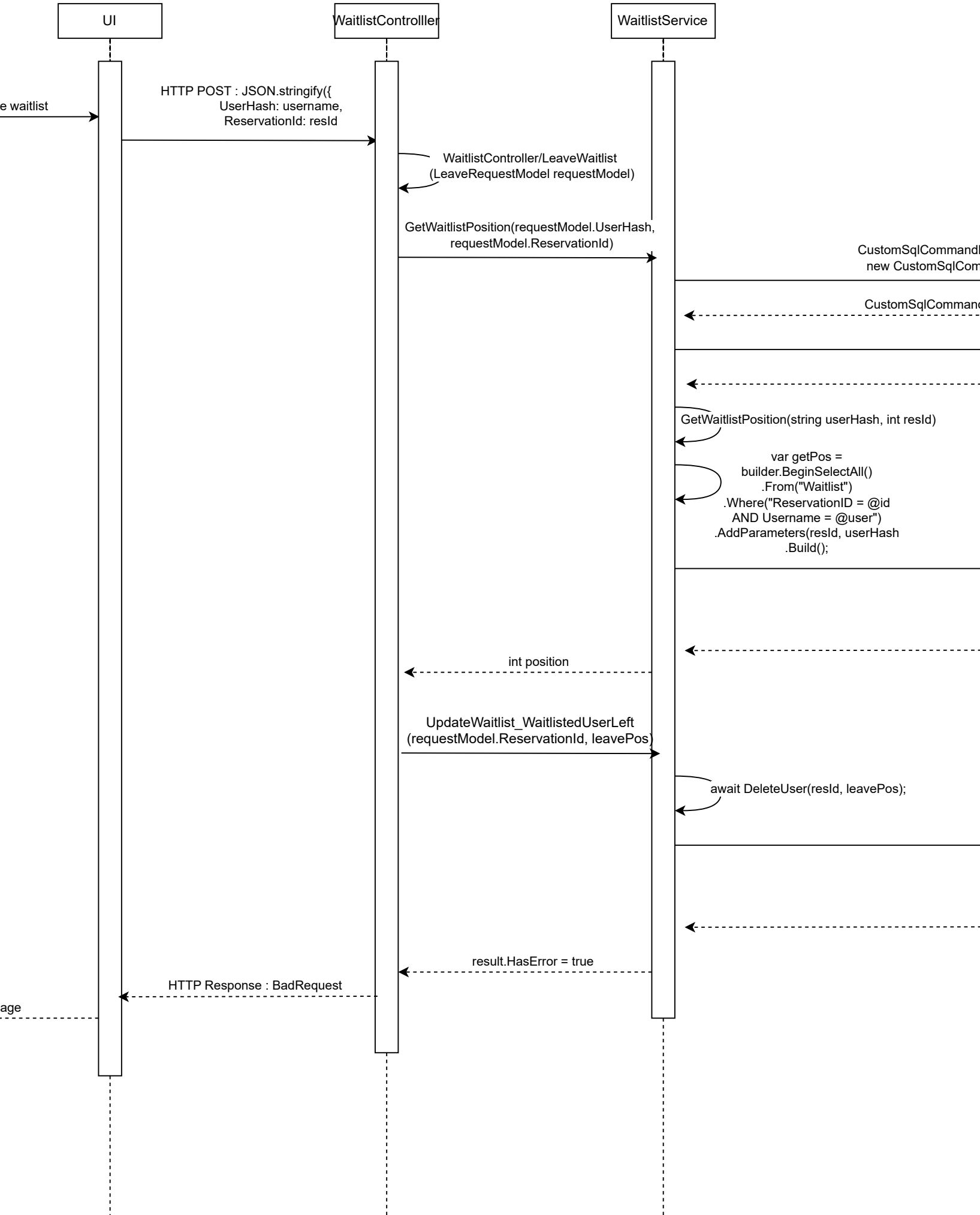

UI
WaitlistControlller
WaitlistService
e waitlist
HTTP POST : JSON.stringify({
UserHash: username,
ReservationId: resId
WaitlistController/LeaveWaitlist
(LeaveRequestModel requestModel)
GetWaitlistPosition(requestModel.UserHash,
requestModel.ReservationId)
CustomSqlCommandI
new CustomSqlCom
CustomSqlCommanc
GetWaitlistPosition(string userHash, int resId)
var getPos =
builder.BeginSelectAll()
.From("Waitlist")
.Where("ReservationID = @id
AND Username = @user")
.AddParameters(resId, userHash
.Build();
int position
UpdateWaitlist_WaitlistedUserLeft
(requestModel.ReservationId, leavePos)
await DeleteUser(resId, leavePos);
result.HasError = false
HTTP Response : 200
lly left waitlist

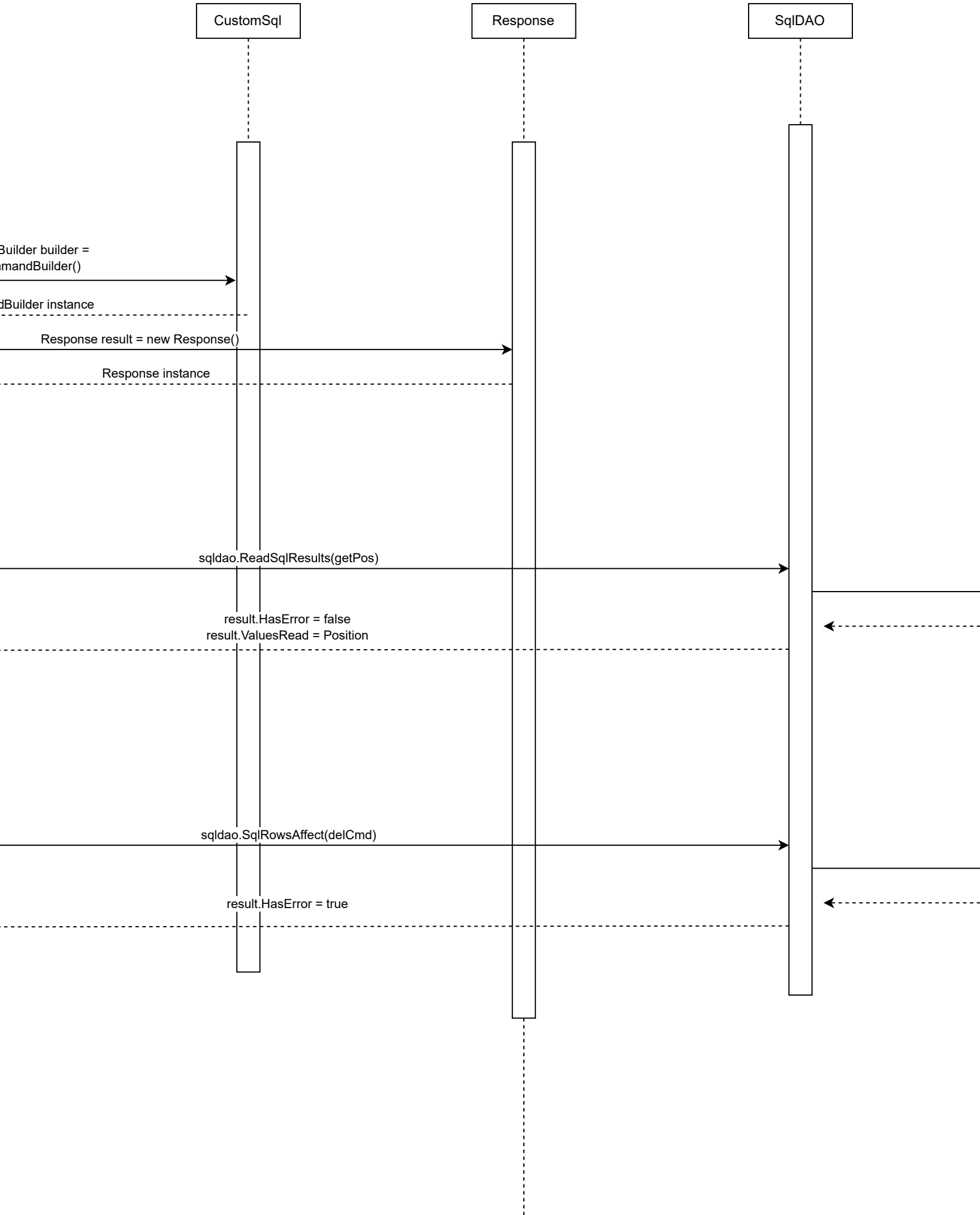

CustomSql
Response
SqlDAO
Builder builder =
mandBuilder()
dBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResults(getPos)
result.HasError = false
result.ValuesRead = Position
sqldao.SqlRowsAffect(delCmd)
result.HasError = false
result.ValuesRead = 0

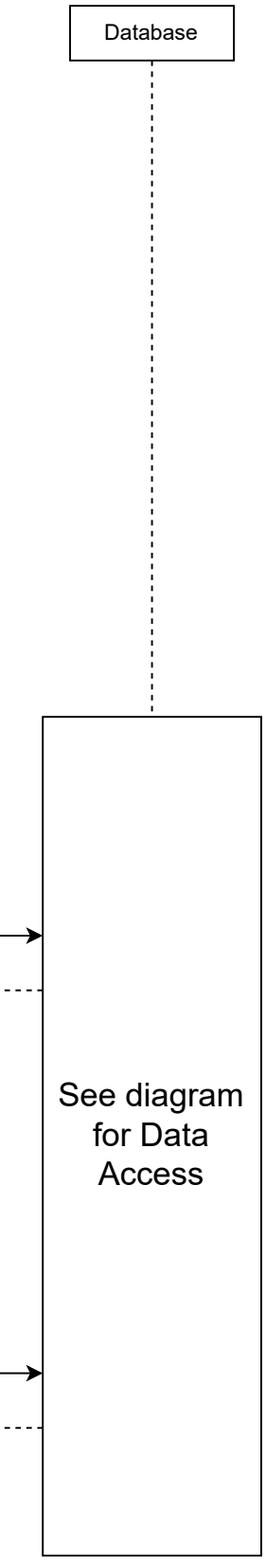

Database
See diagram for Data Access

Leave Waitlist - Failure


User
Request to leav
Error Mess

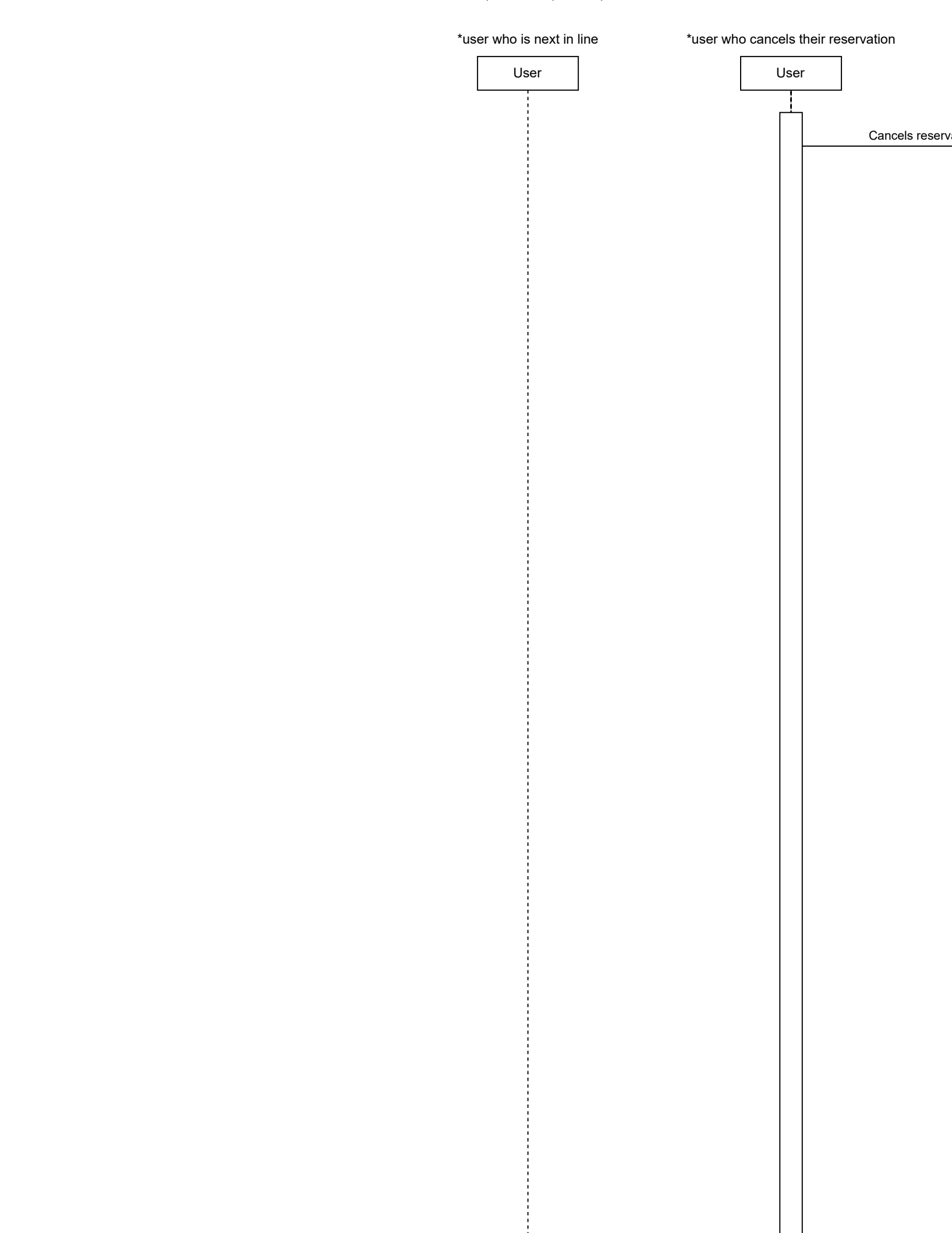

UI
WaitlistControlller
WaitlistService
e waitlist
HTTP POST : JSON.stringify({
UserHash: username,
ReservationId: resId
WaitlistController/LeaveWaitlist
(LeaveRequestModel requestModel)
GetWaitlistPosition(requestModel.UserHash,
requestModel.ReservationId)
CustomSqlCommandI
new CustomSqlCom
CustomSqlComman
GetWaitlistPosition(string userHash, int resId)
var getPos =
builder.BeginSelectAll()
.From("Waitlist")
.Where("ReservationID = @id
AND Username = @user")
.AddParameters(resId, userHash
.Build();
int position
UpdateWaitlist_WaitlistedUserLeft
(requestModel.ReservationId, leavePos)
await DeleteUser(resId, leavePos);
result.HasError = true
HTTP Response : BadRequest
age

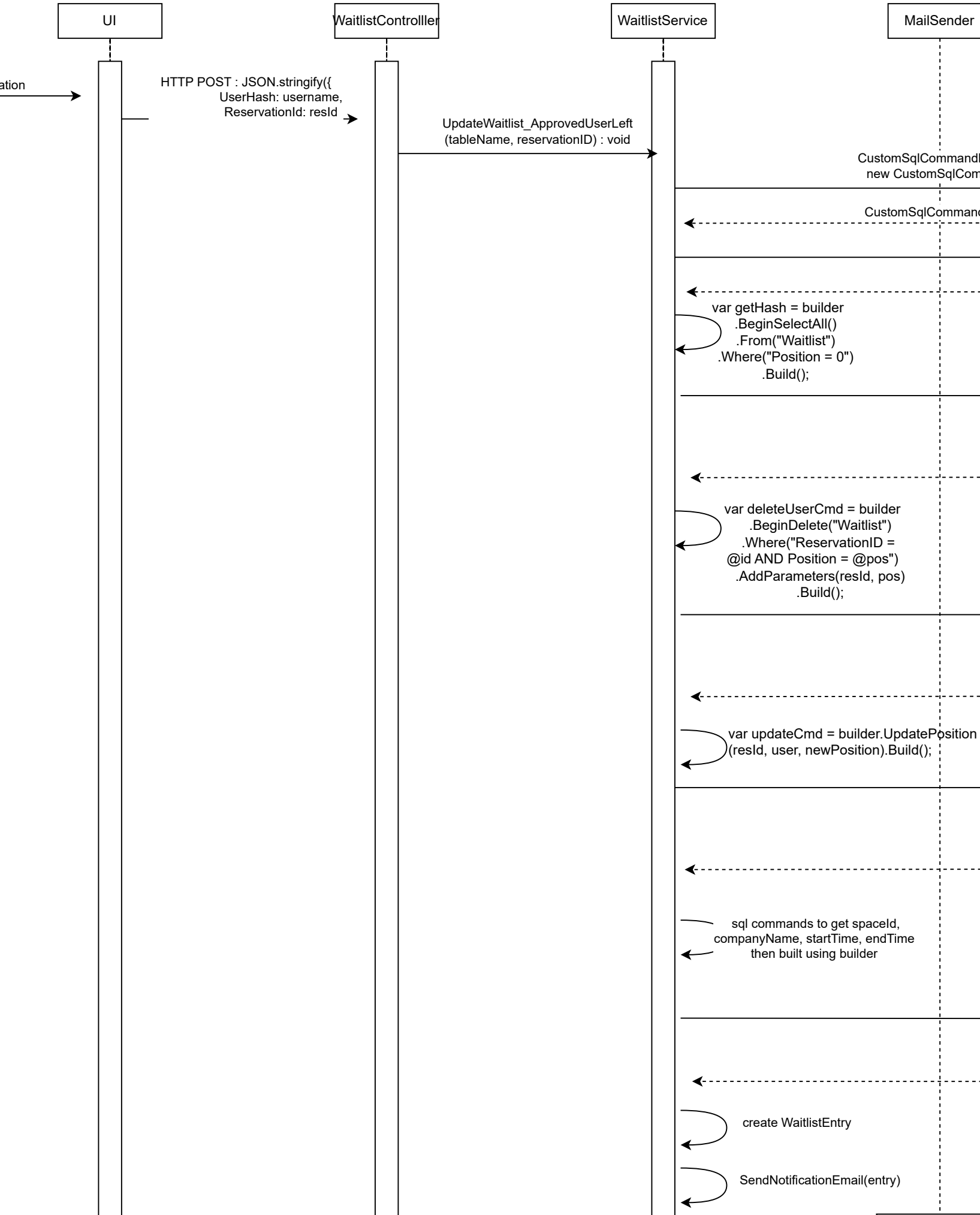

CustomSql
Response
SqlDAO
Builder builder =
mandBuilder()
dBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResults(getPos)
result.HasError = false
result.ValuesRead = Position
sqldao.SqlRowsAffect(delCmd)
result.HasError = true

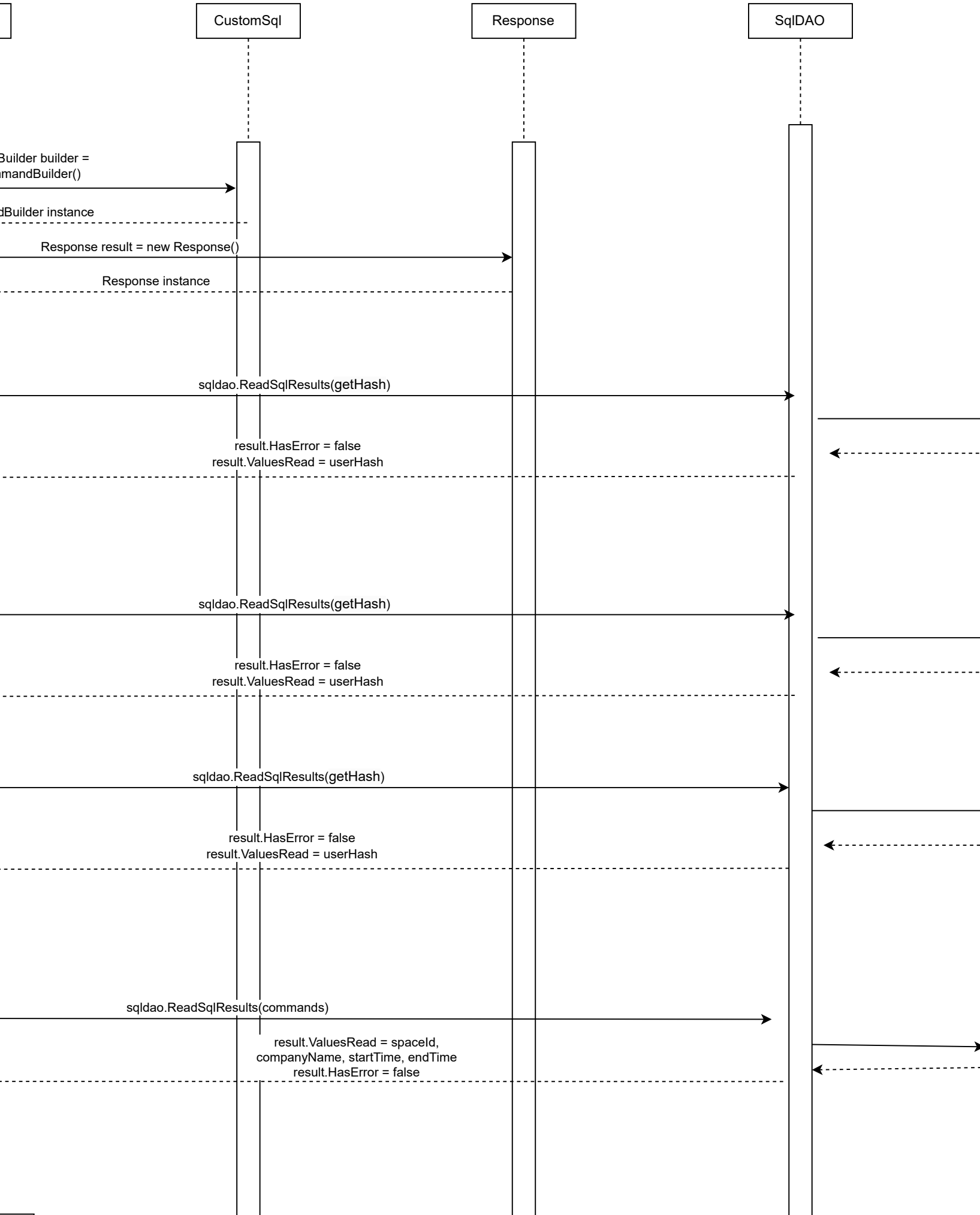

Database
See diagram for Data Access

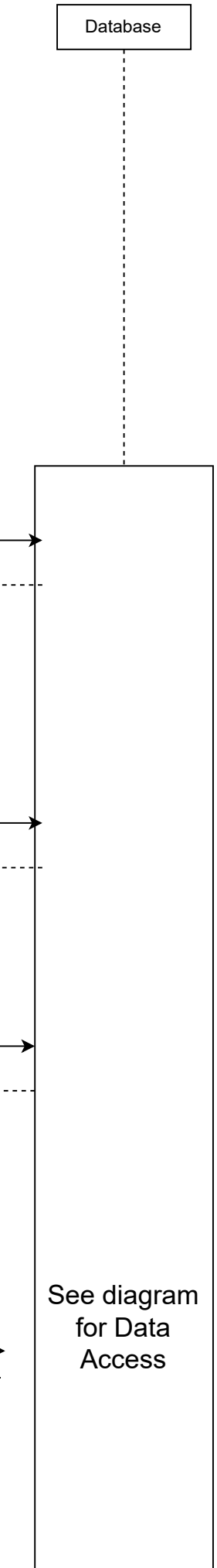
Leave Waitlist - Top of waitlist leaves

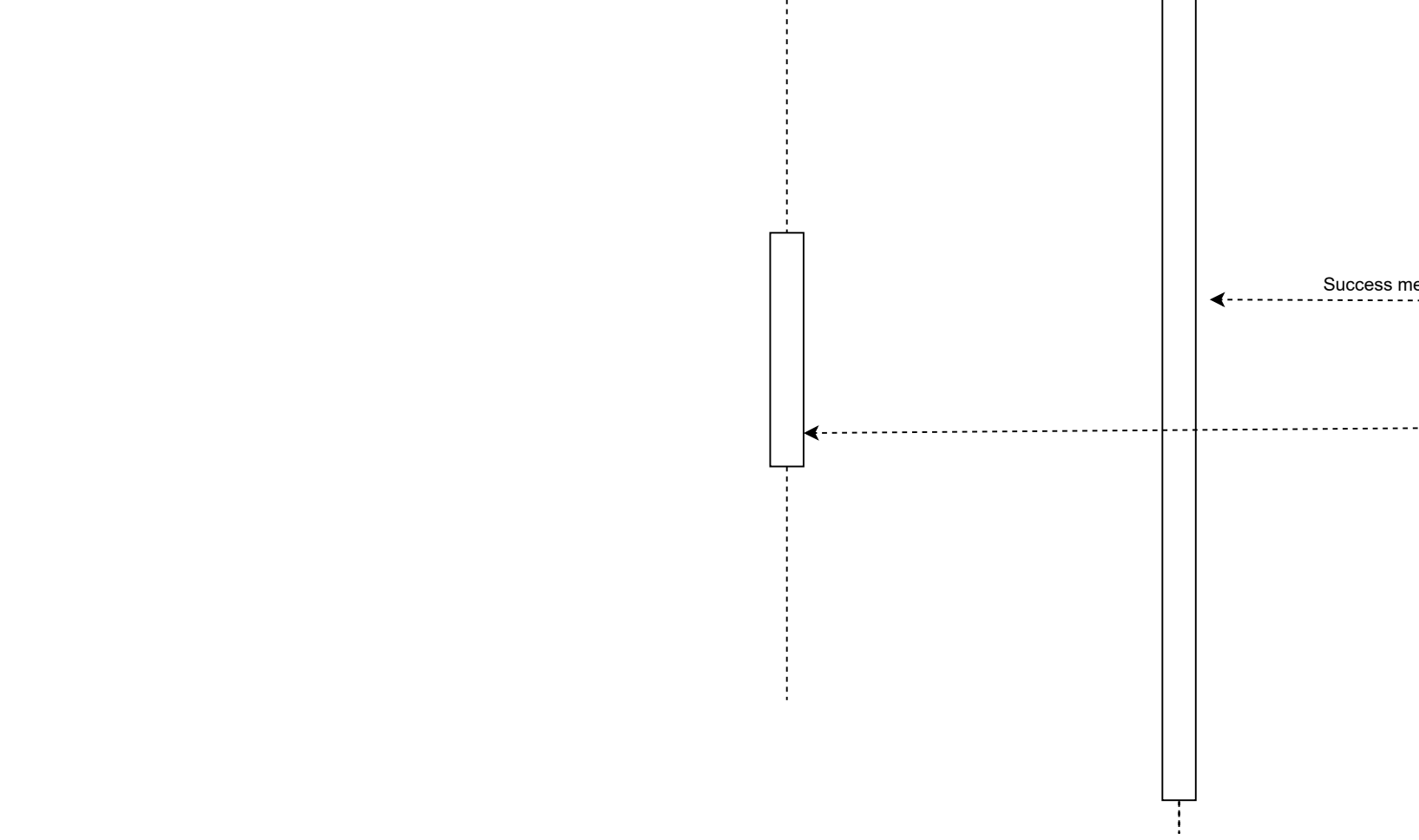
When a user reserves a space, they will be added to the top of the waitlist for that space and time. If they cancel their reservation, they will be removed from the waitlist, which will allow for the next user on the list to be notified that the space has opened up.

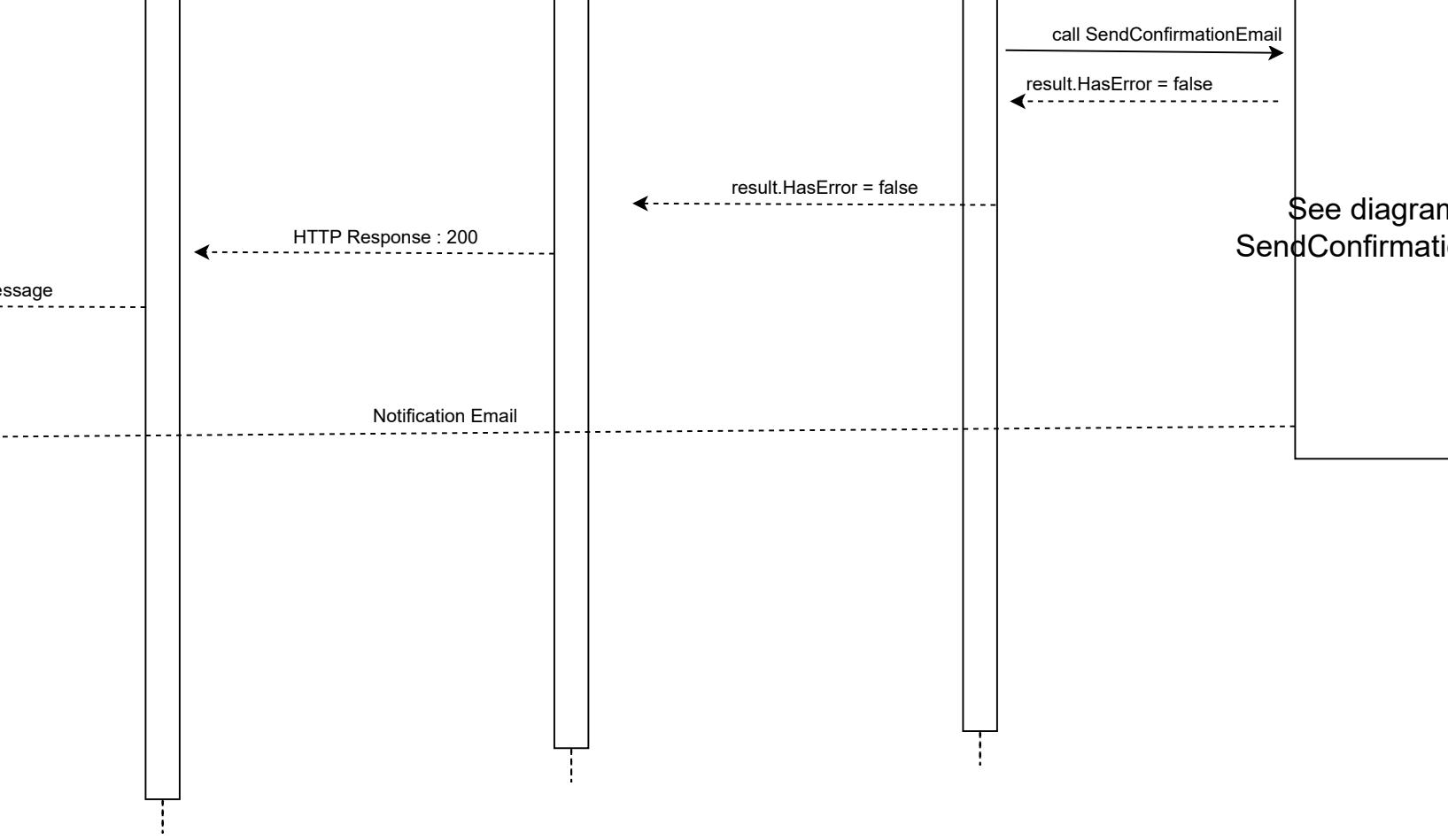

*user who is next in line
User
*user who cancels their reservation
User
Cancels reserva

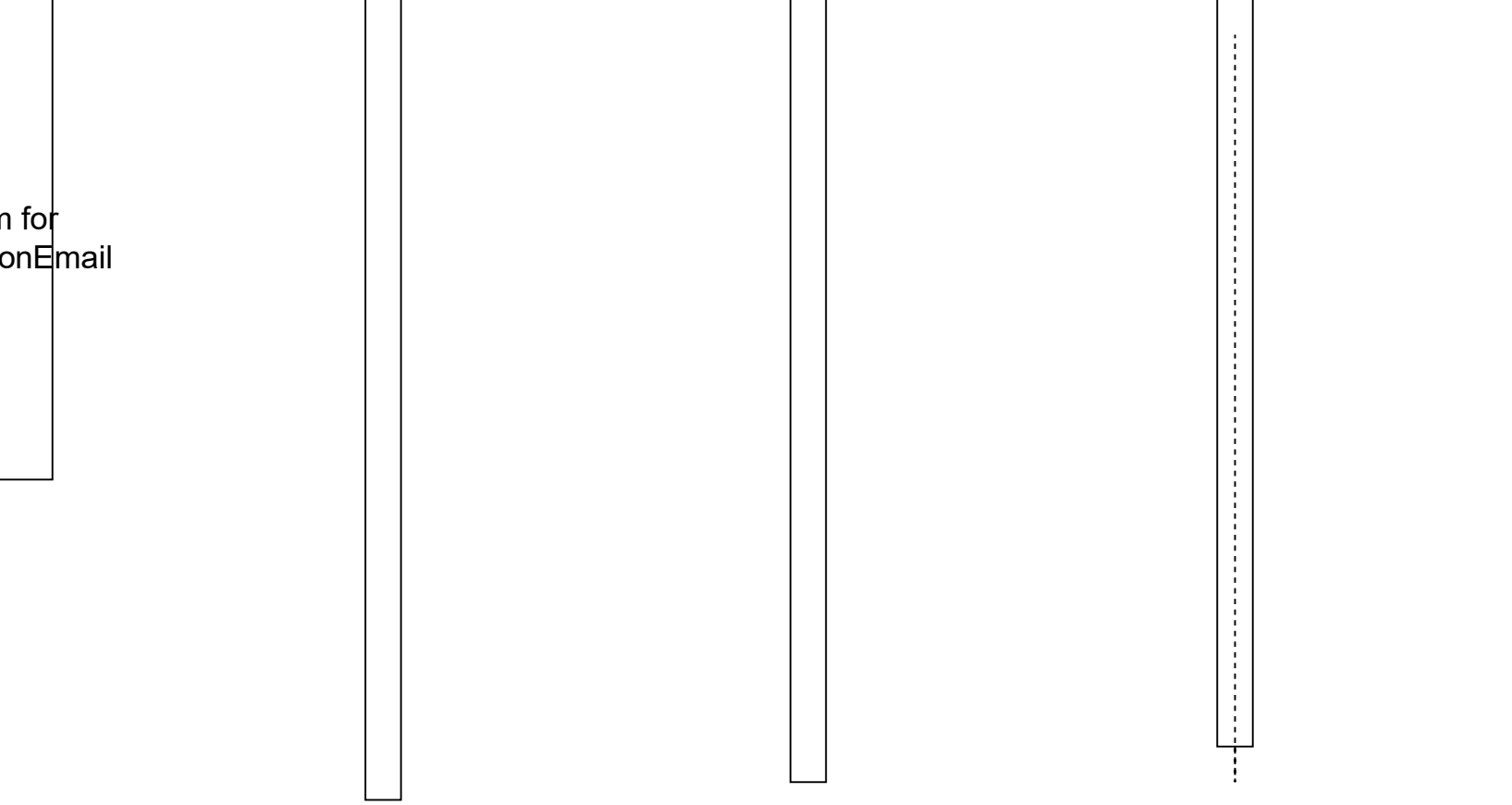

UI
WaitlistControlller
WaitlistService
MailSender
ation
HTTP POST : JSON.stringify({
UserHash: username,
ReservationId: resId
UpdateWaitlist_ApprovedUserLeft
(tableName, reservationID) : void
CustomSqlCommandI
new CustomSqlCom
CustomSqlCommanc
var getHash = builder
.BeginSelectAll()
.From("Waitlist")
.Where("Position = 0")
.Build();
var deleteUserCmd = builder
.BeginDelete("Waitlist")
.Where("ReservationID =
@id AND Position = @pos")
.AddParameters(resId, pos)
.Build();
var updateCmd = builder.UpdatePosition
(resId, user, newPosition).Build();
sql commands to get spaceId,
companyName, startTime, endTime
then built using builder
create WaitlistEntry
SendNotificationEmail(entry)

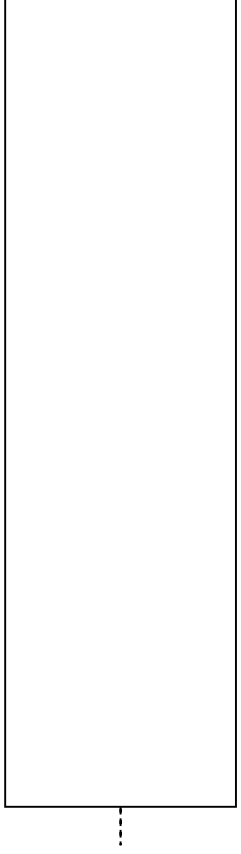

CustomSql
Response
SqlDAO
Builder builder =
mandBuilder()
dBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResults(getHash)
result.HasError = false
result.ValuesRead = userHash
sqldao.ReadSqlResults(getHash)
result.HasError = false
result.ValuesRead = userHash
sqldao.ReadSqlResults(getHash)
result.HasError = false
result.ValuesRead = userHash
sqldao.ReadSqlResults(commands)
result.ValuesRead = spaceId,
companyName, startTime, endTime
result.HasError = false


Database
See diagram for Data Access

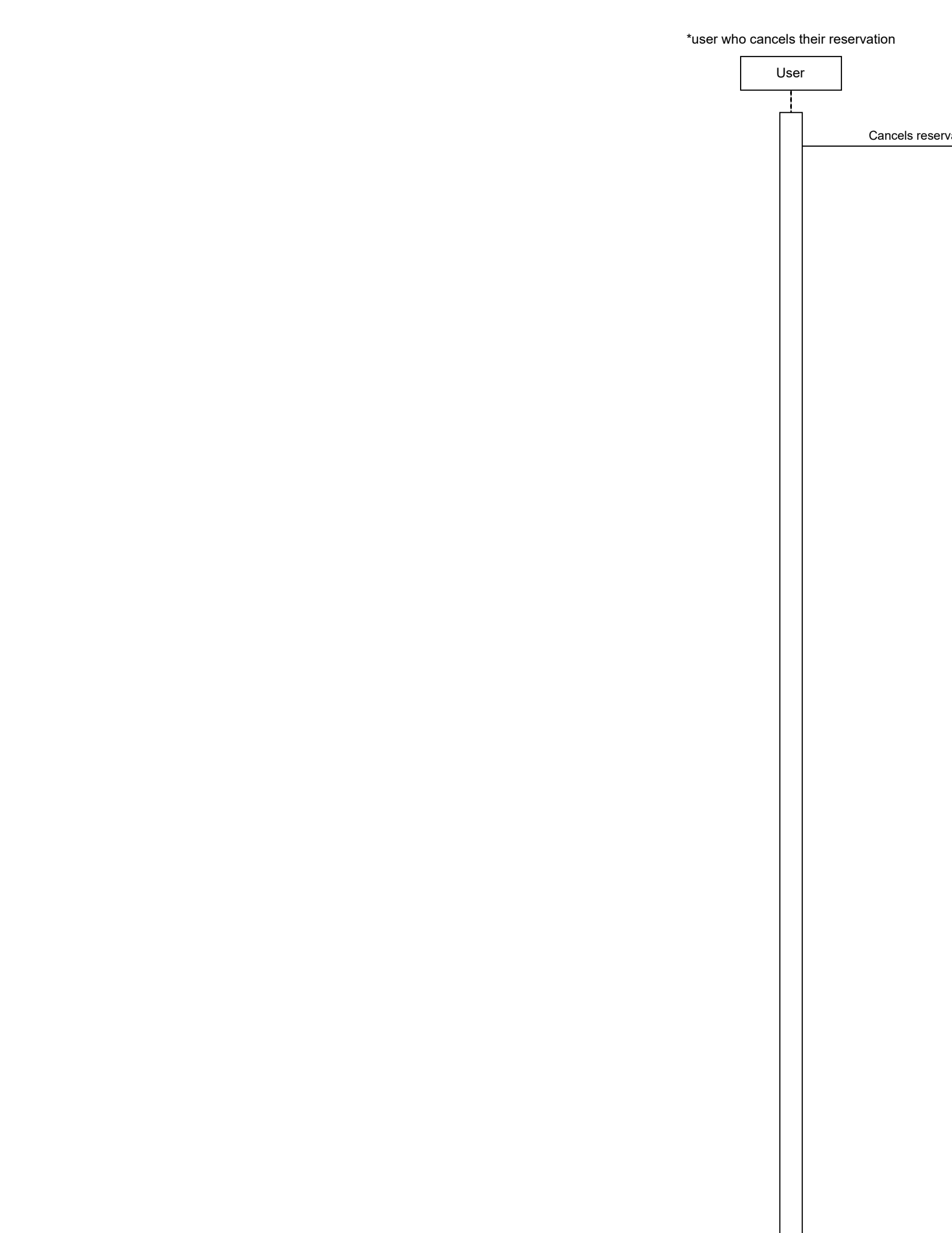

Success me

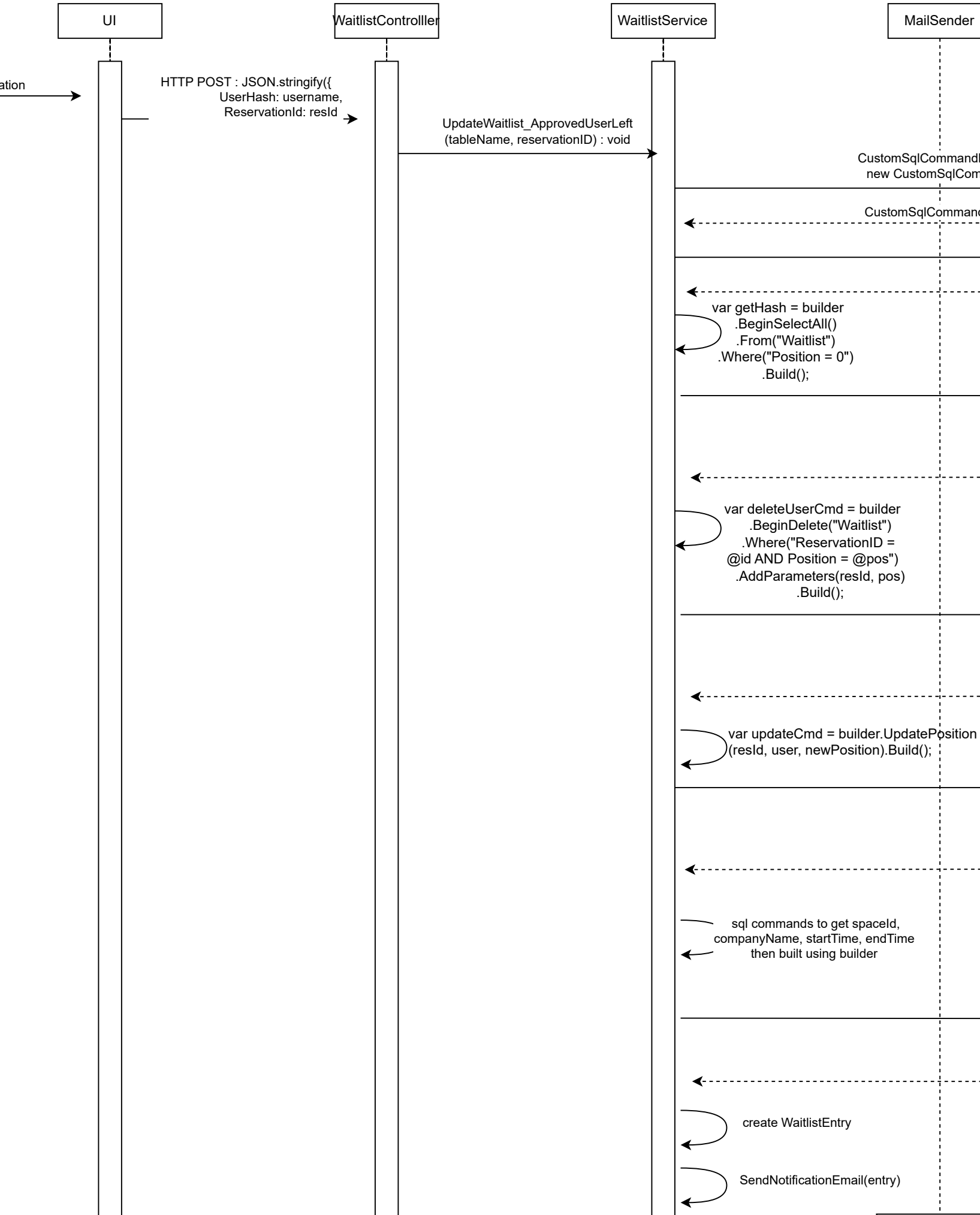

call SendConfirmationEmail
result.HasError = false
result.HasError = false
See diagram
SendConfirmati
HTTP Response : 200
essage
Notification Email

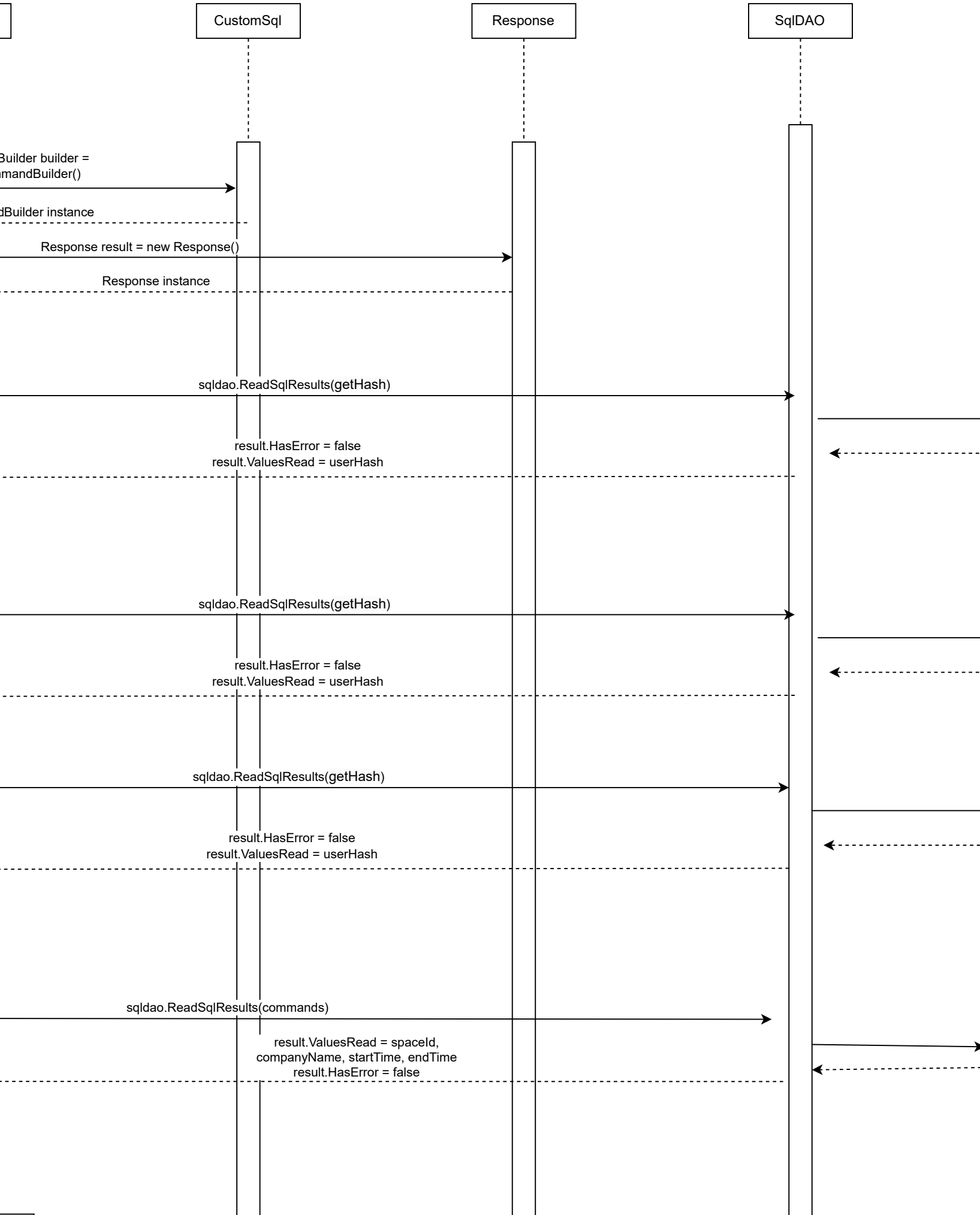

n for
onEmail

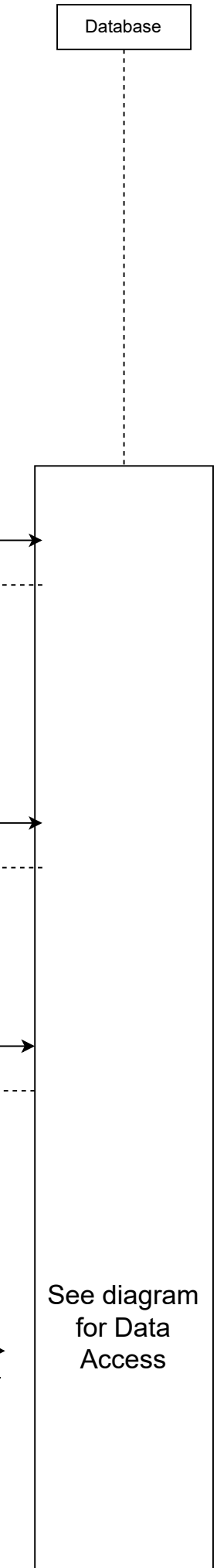


Leave Waitlist - Top of waitlist leaves - Failure

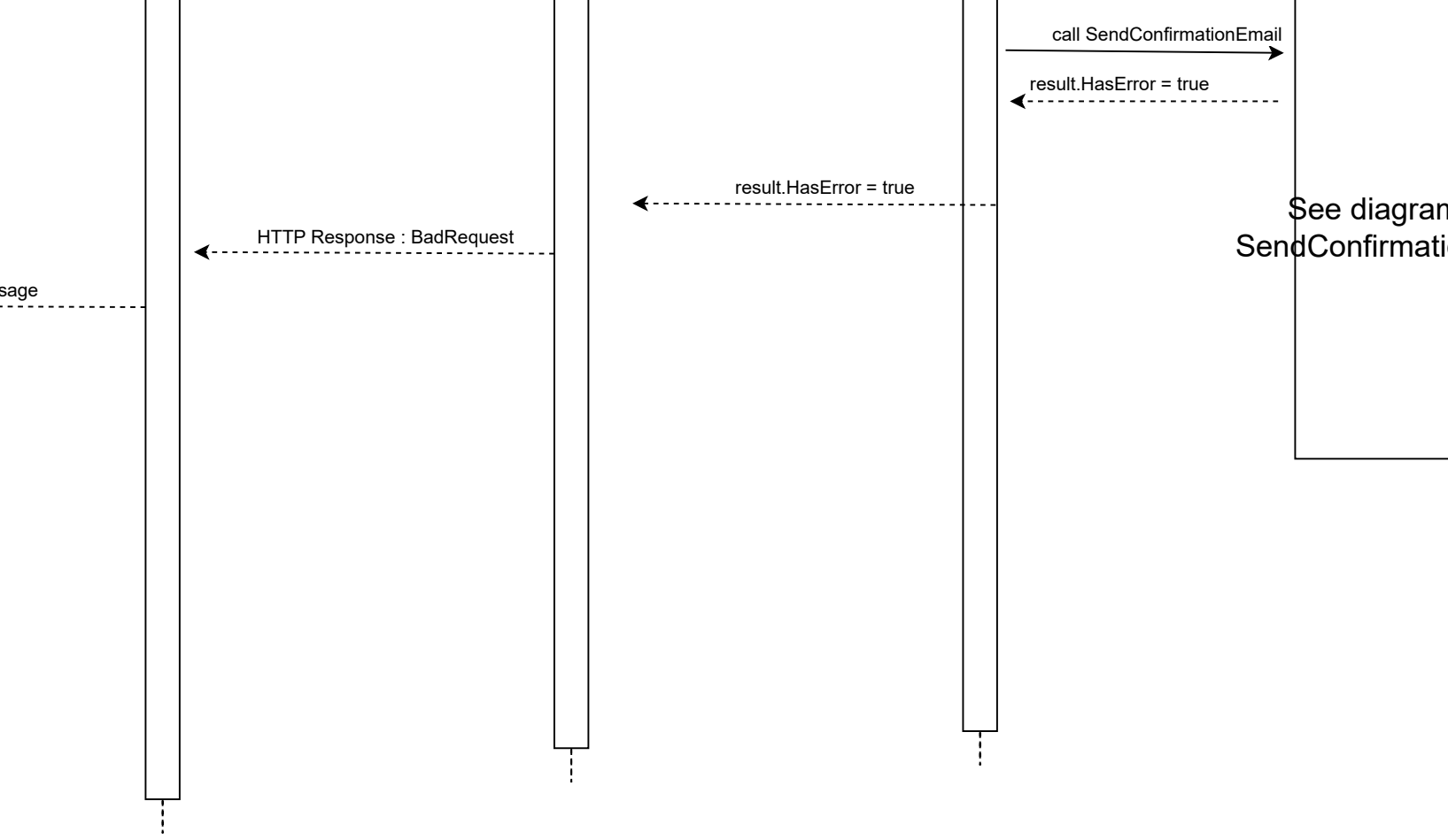

*user who cancels their reservation
User
Cancels reserva

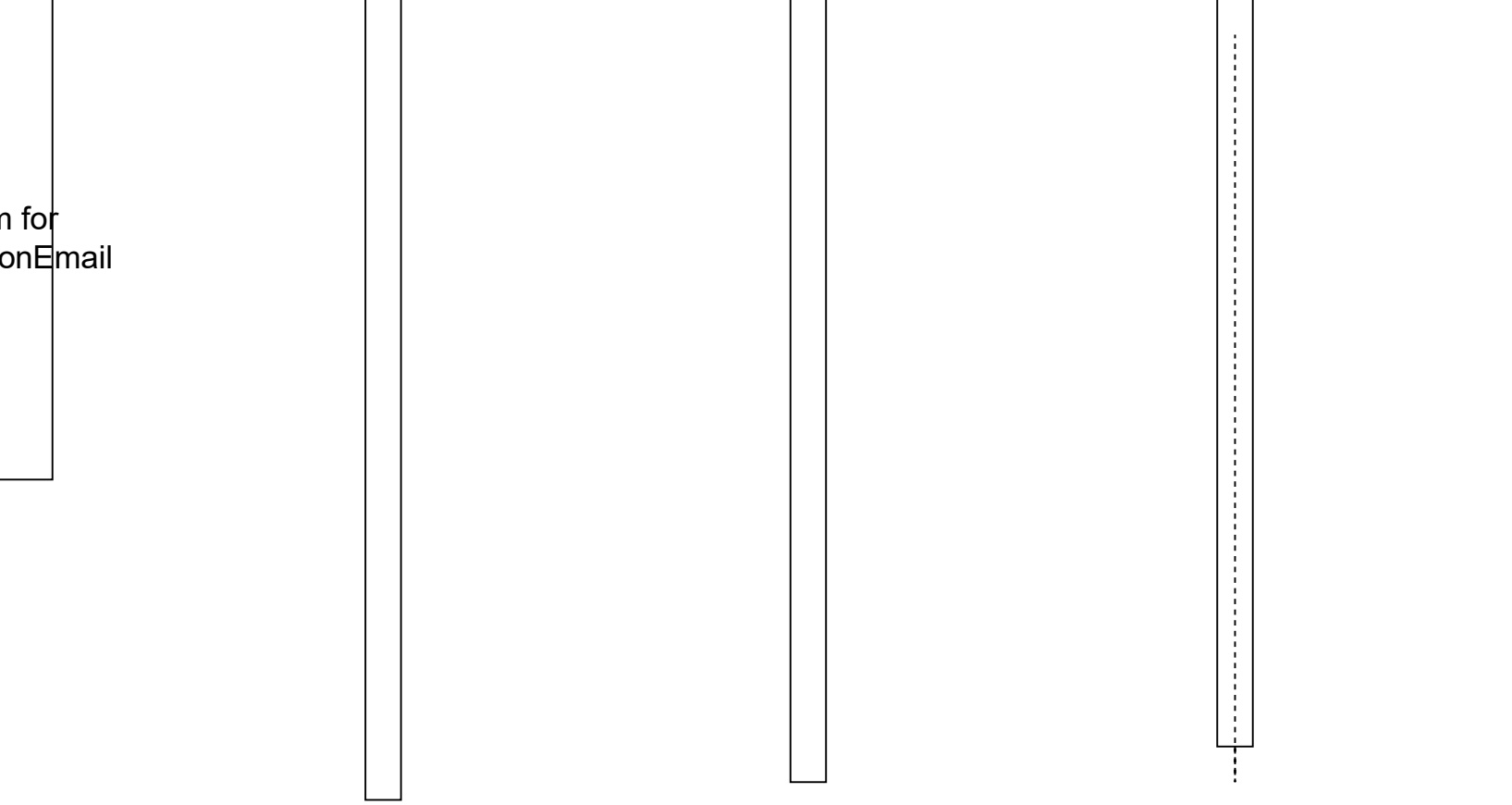

UI
WaitlistControlller
WaitlistService
MailSender
ation
HTTP POST : JSON.stringify({
UserHash: username,
ReservationId: resId
UpdateWaitlist_ApprovedUserLeft
(tableName, reservationID) : void
CustomSqlCommandI
new CustomSqlCom
CustomSqlCommanc
var getHash = builder
.BeginSelectAll()
.From("Waitlist")
.Where("Position = 0")
.Build();
var deleteUserCmd = builder
.BeginDelete("Waitlist")
.Where("ReservationID =
@id AND Position = @pos")
.AddParameters(resId, pos)
.Build();
var updateCmd = builder.UpdatePosition
(resId, user, newPosition).Build();
sql commands to get spaceId,
companyName, startTime, endTime
then built using builder
create WaitlistEntry
SendNotificationEmail(entry)

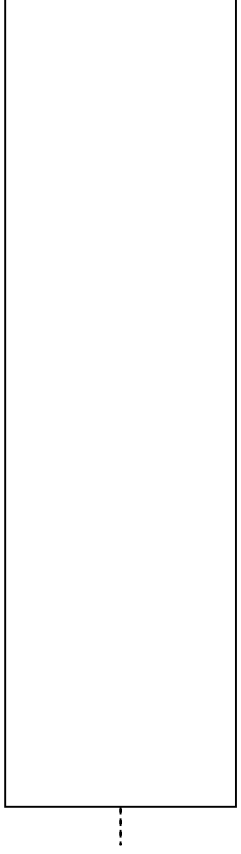

CustomSql
Response
SqlDAO
Builder builder =
mandBuilder()
dBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResults(getHash)
result.HasError = false
result.ValuesRead = userHash
sqldao.ReadSqlResults(getHash)
result.HasError = false
result.ValuesRead = userHash
sqldao.ReadSqlResults(getHash)
result.HasError = false
result.ValuesRead = userHash
sqldao.ReadSqlResults(commands)
result.ValuesRead = spaceId,
companyName, startTime, endTime
result.HasError = false


Database
See diagram for Data Access

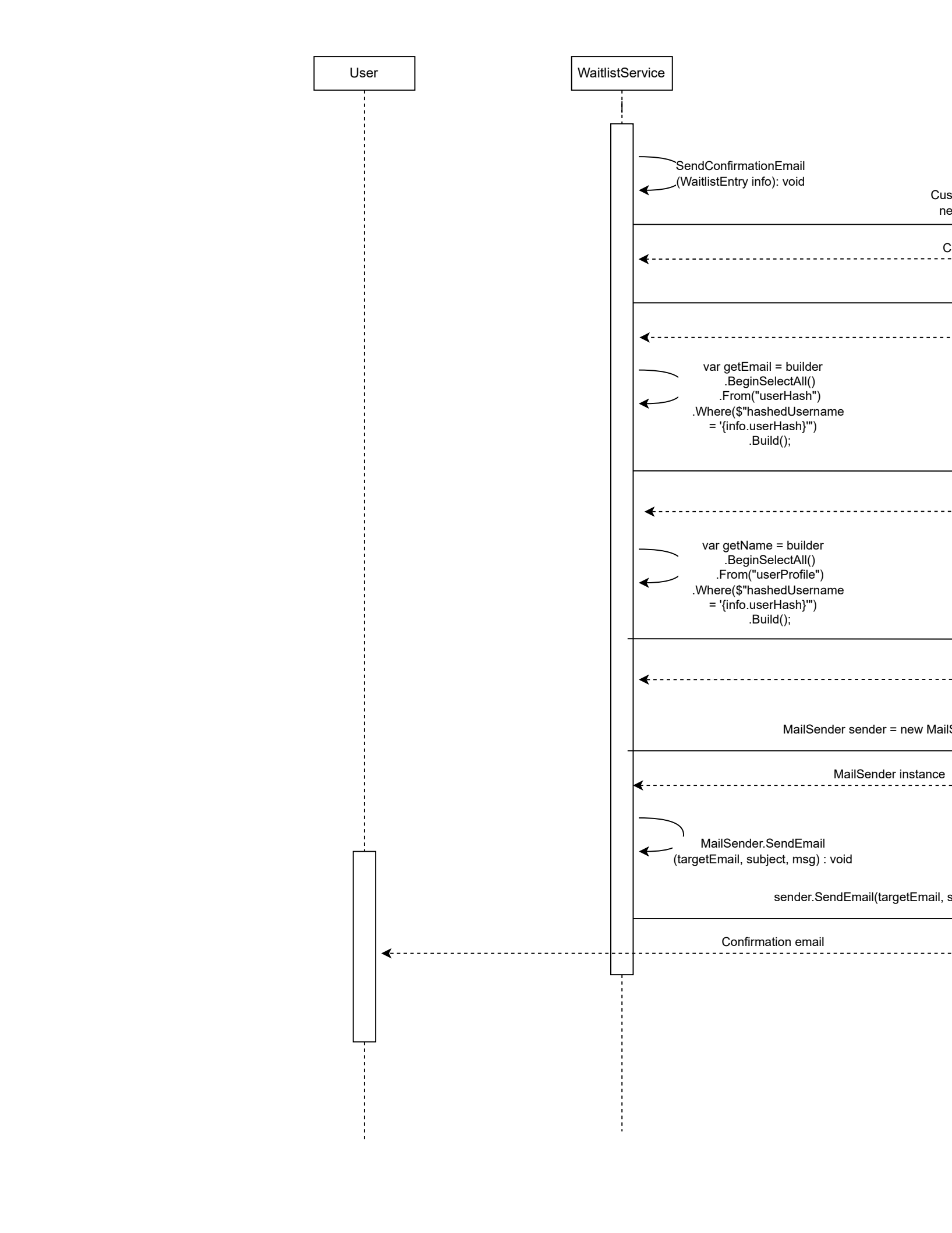
Error mes

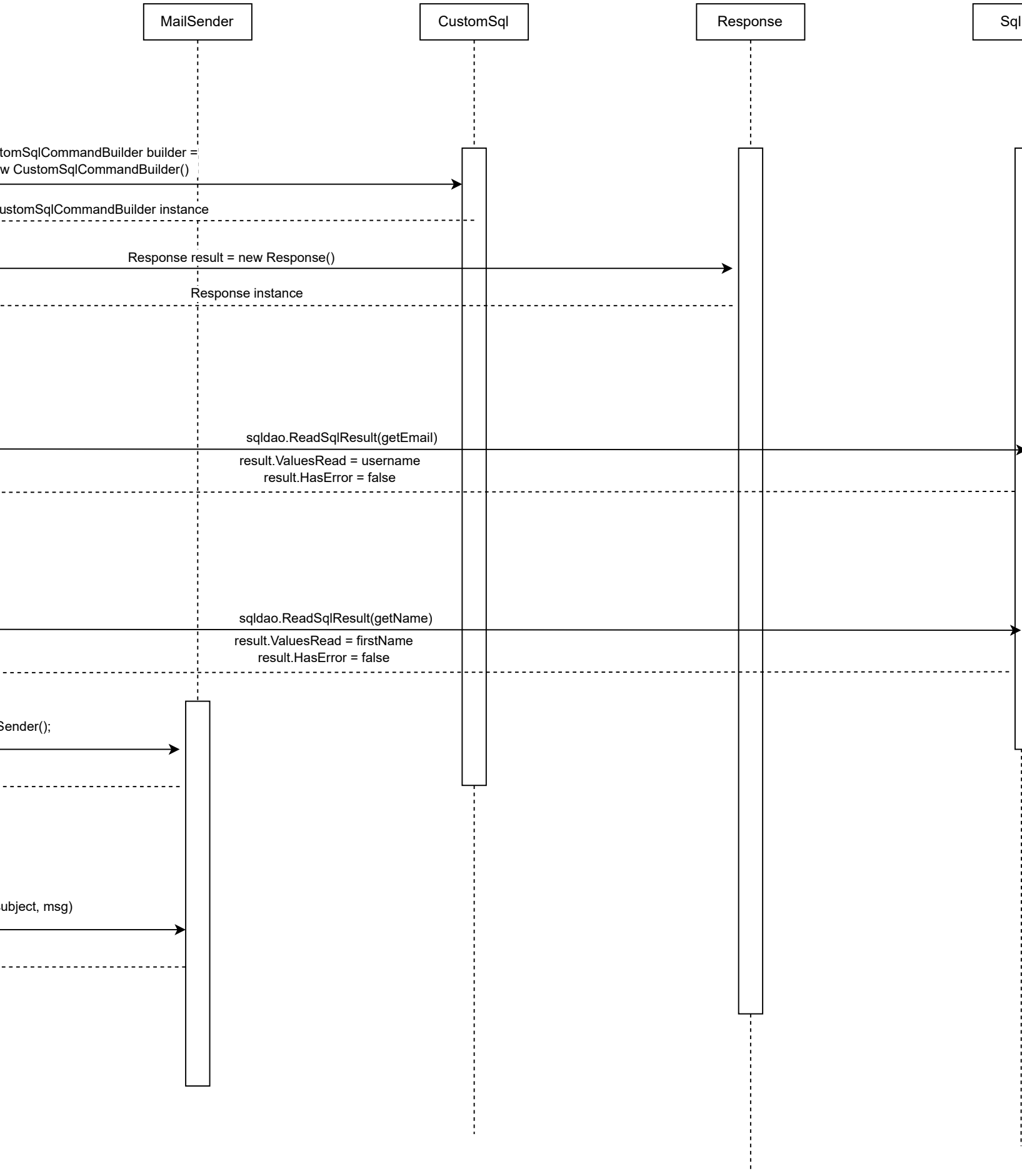

call SendConfirmationEmail
result.HasError = true
result.HasError = true
See diagram
SendConfirmati
HTTP Response : BadRequest
sage

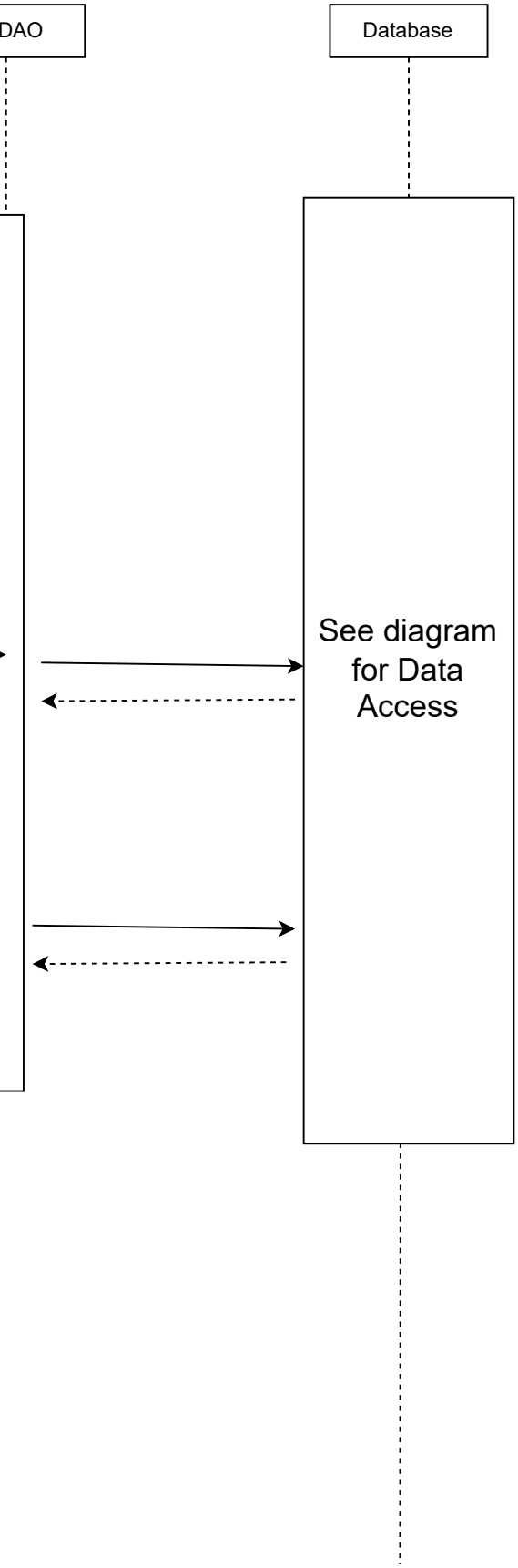

n for
onEmail



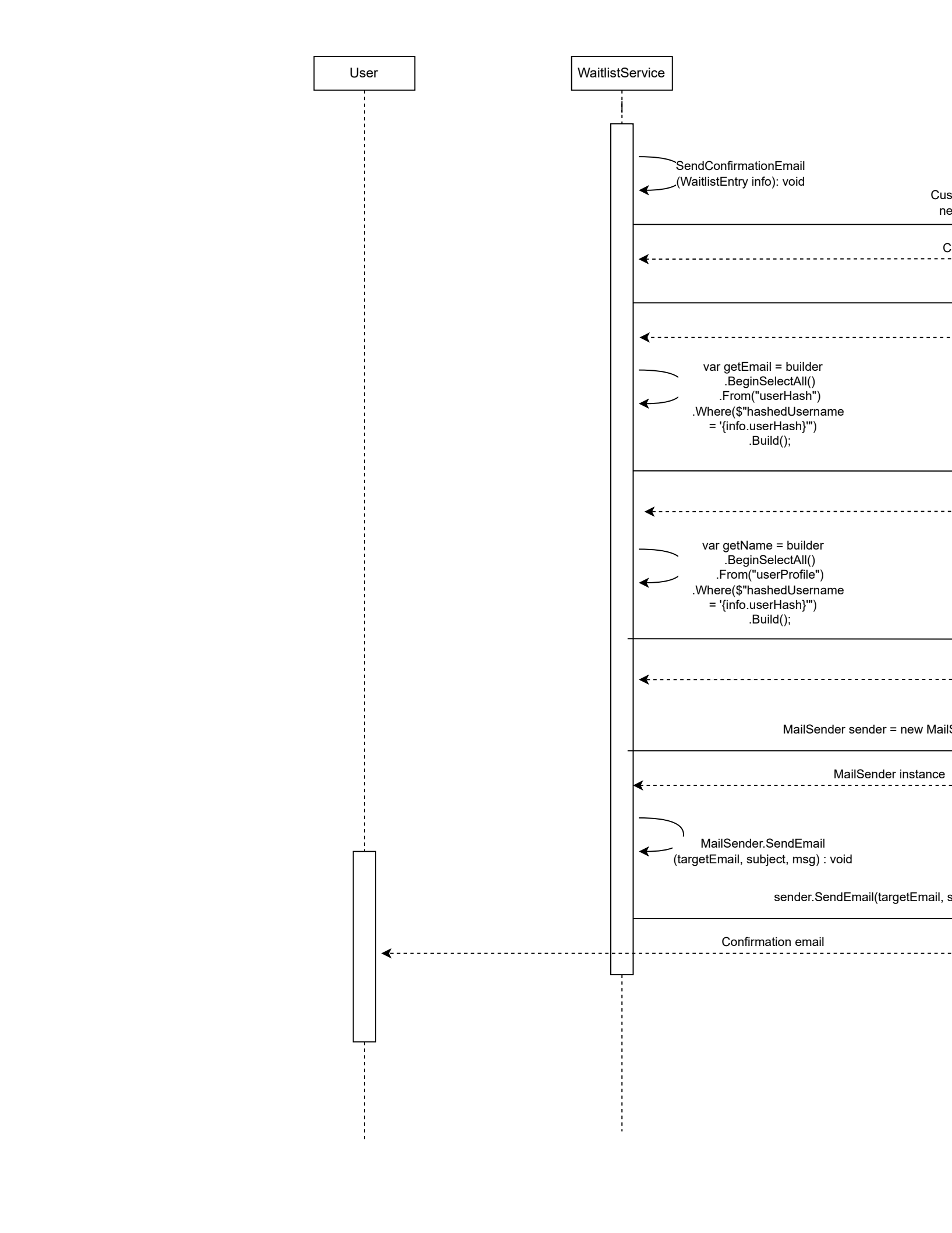
Confirmation Email - Success

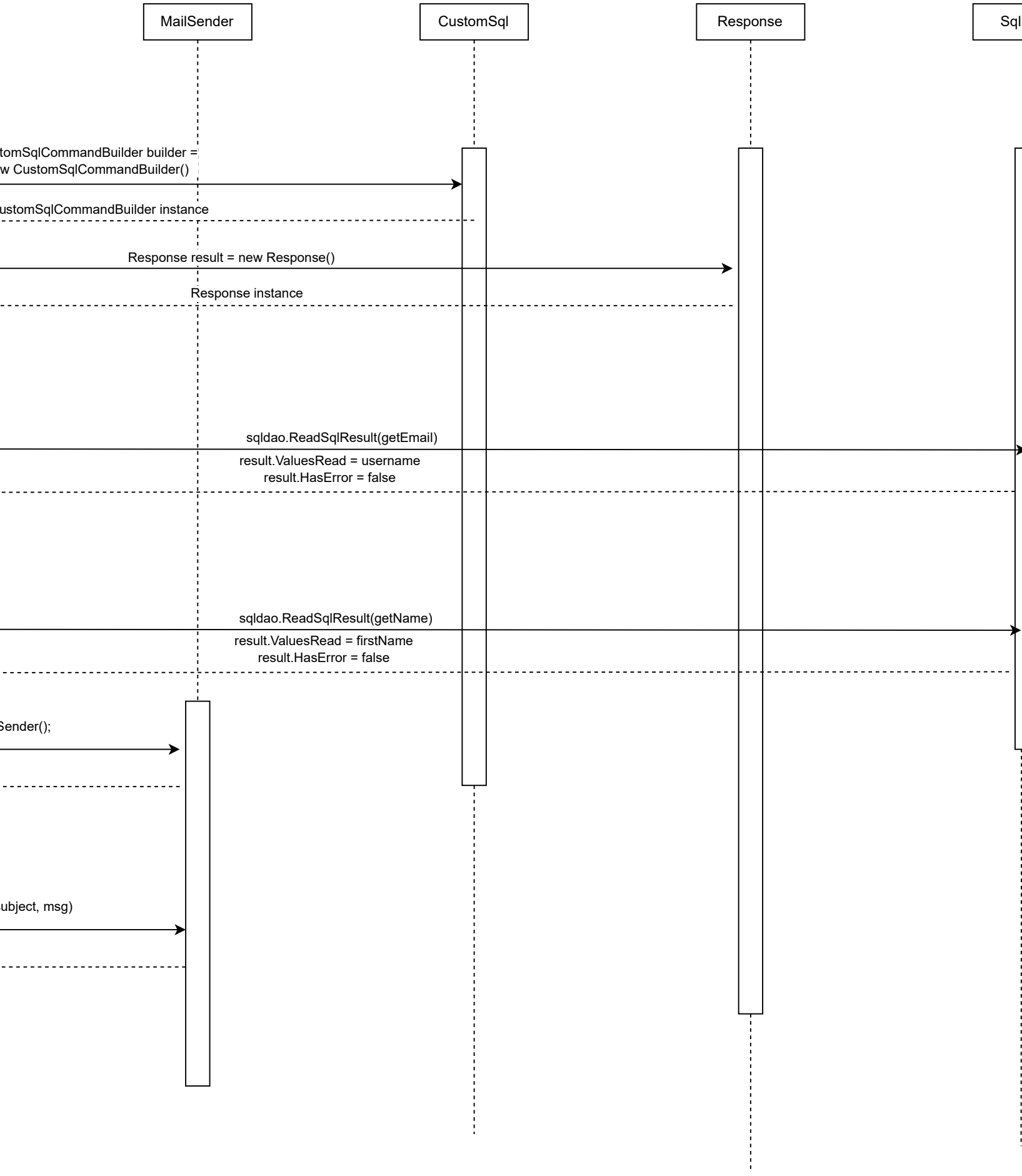

User
WaitlistService
SendConfirmationEmail
(WaitlistEntry info): void
Cus
ne
C
var getEmail = builder
.BeginSelectAll()
.From("userHash")
.Where($"hashedUsername
= '{info.userHash}'")
.Build();
var getName = builder
.BeginSelectAll()
.From("userProfile")
.Where($"hashedUsername
= '{info.userHash}'")
.Build();
MailSender sender = new Mail
MailSender instance
MailSender.SendEmail
(targetEmail, subject, msg) : void
sender.SendEmail(targetEmail, s
Confirmation email

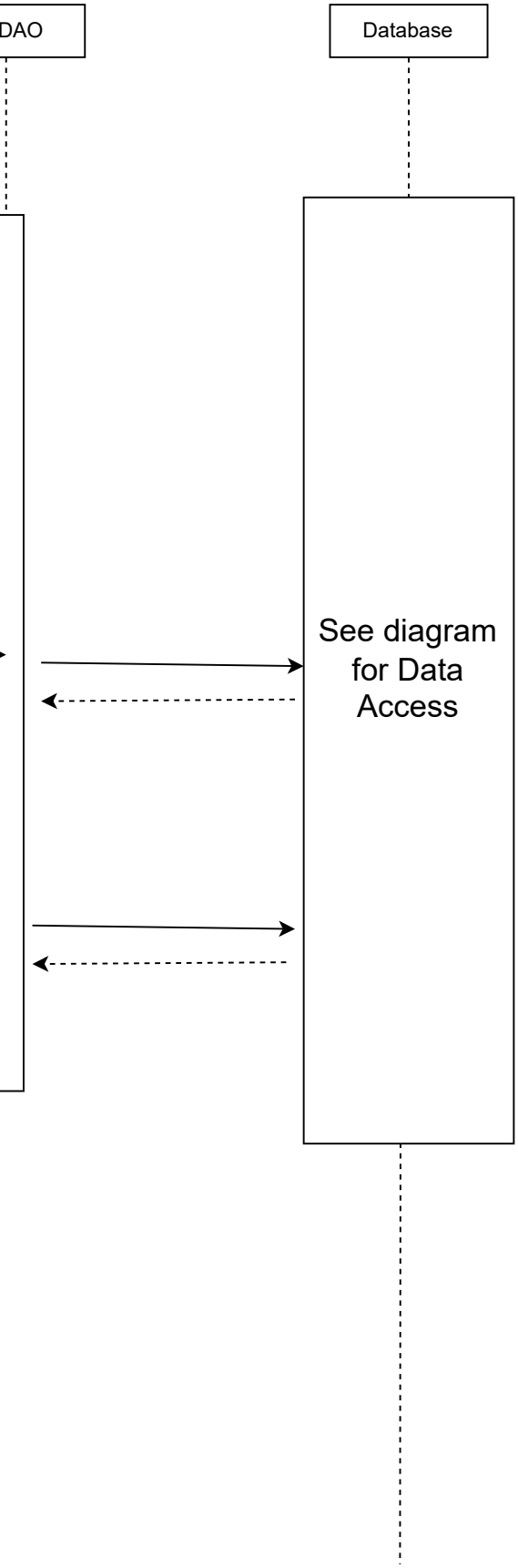

MailSender
CustomSql
Response
Sql
tomSqlCommandBuilder builder =
w CustomSqlCommandBuilder()
ustomSqlCommandBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResult(getEmail)
result.ValuesRead = username
result.HasError = false
sqldao.ReadSqlResult(getName)
result.ValuesRead = firstName
result.HasError = false
Sender();
ubject, msg)


DAO
Database
See diagram for Data Access

Notification Email - Success

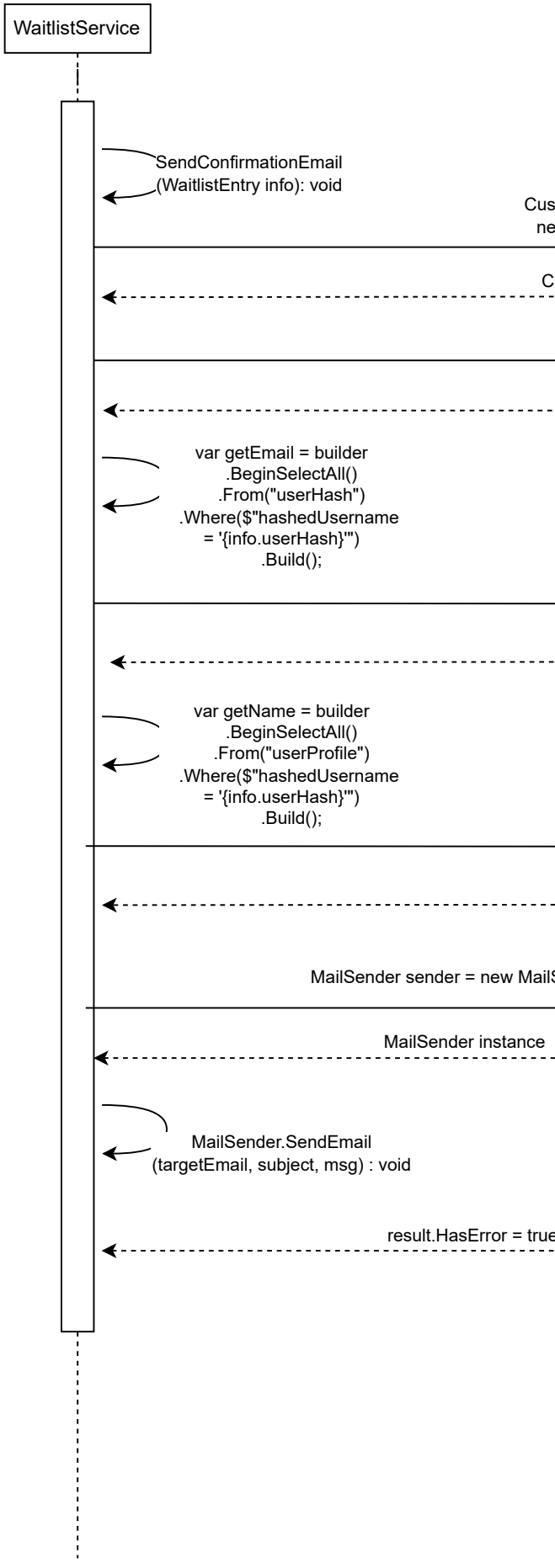

User
WaitlistService
SendConfirmationEmail
(WaitlistEntry info): void
Cus
ne
C
var getEmail = builder
.BeginSelectAll()
.From("userHash")
.Where($"hashedUsername
= '{info.userHash}'")
.Build();
var getName = builder
.BeginSelectAll()
.From("userProfile")
.Where($"hashedUsername
= '{info.userHash}'")
.Build();
MailSender sender = new Mail
MailSender instance
MailSender.SendEmail
(targetEmail, subject, msg) : void
sender.SendEmail(targetEmail, s
Confirmation email

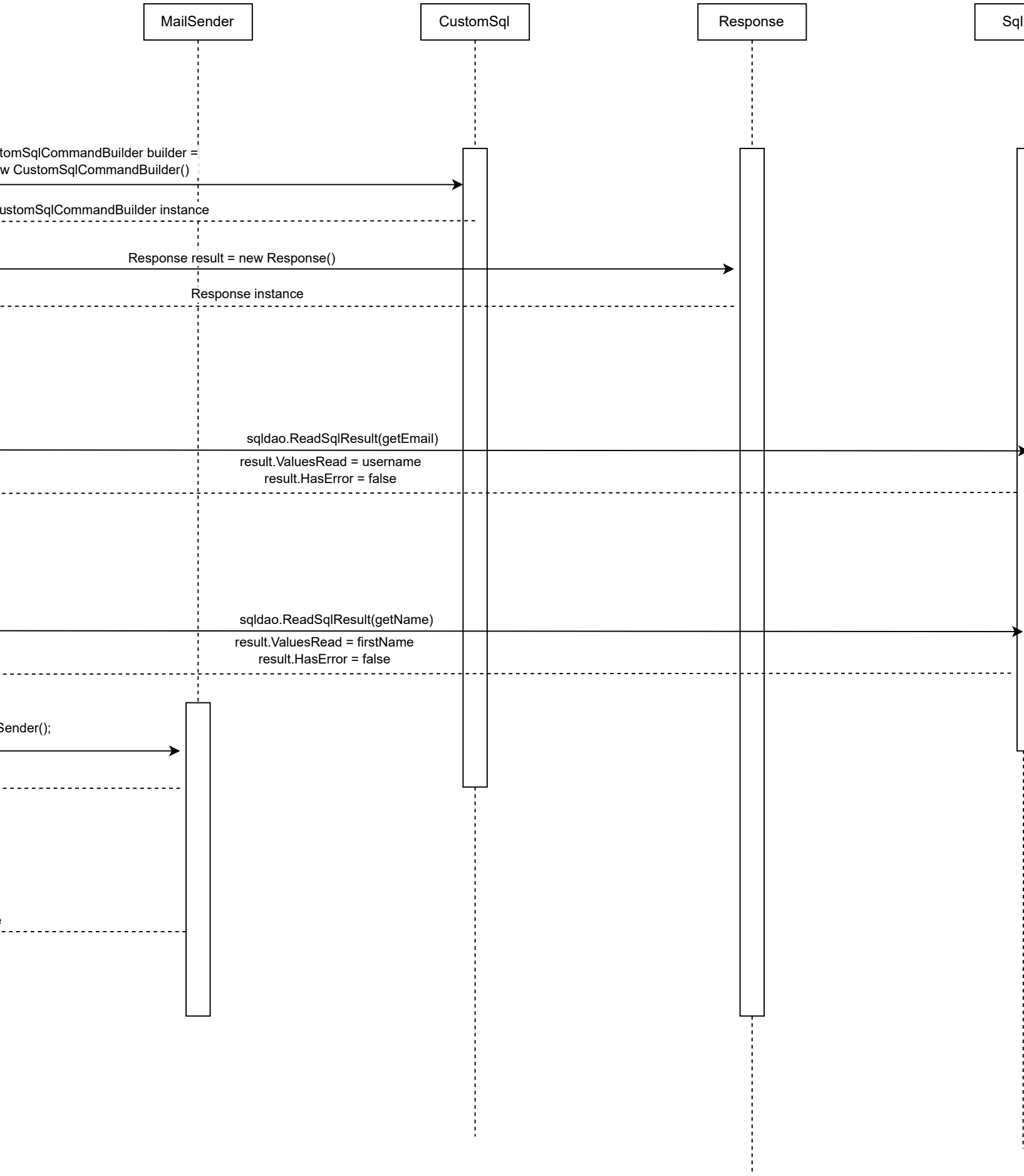

MailSender
CustomSql
Response
Sql
tomSqlCommandBuilder builder =
w CustomSqlCommandBuilder()
ustomSqlCommandBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResult(getEmail)
result.ValuesRead = username
result.HasError = false
sqldao.ReadSqlResult(getName)
result.ValuesRead = firstName
result.HasError = false
Sender();
ubject, msg)

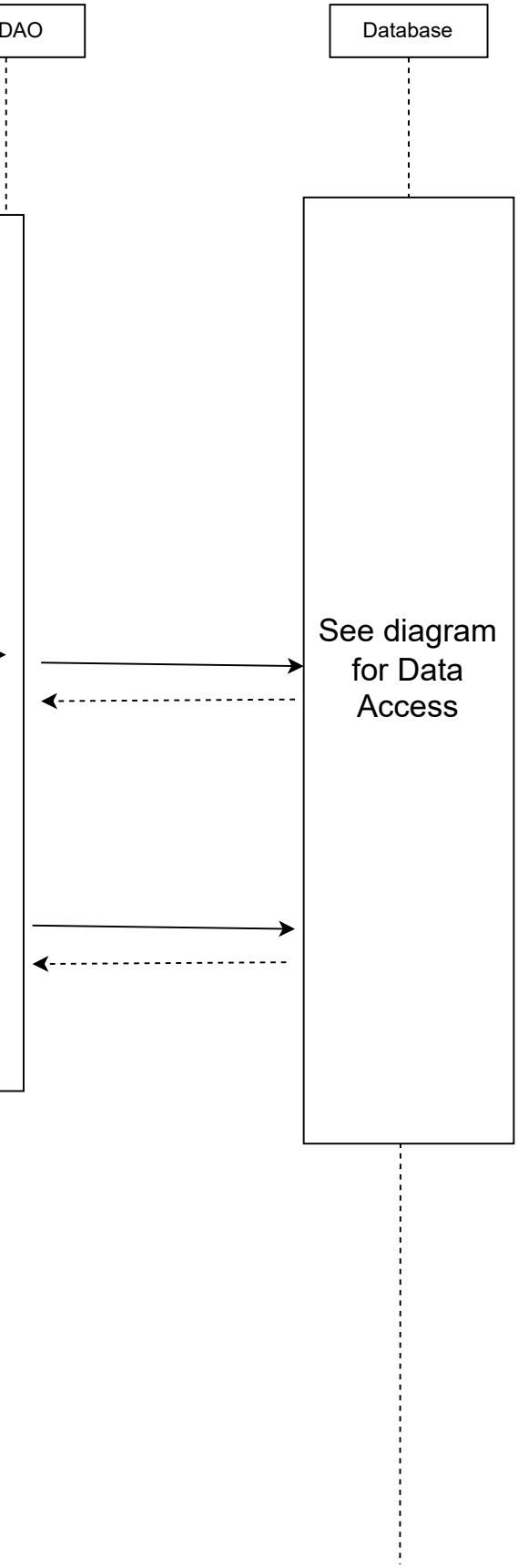

DAO
Database
See diagram for Data Access

Confirmation Email - Success

Confirmation Email - Success

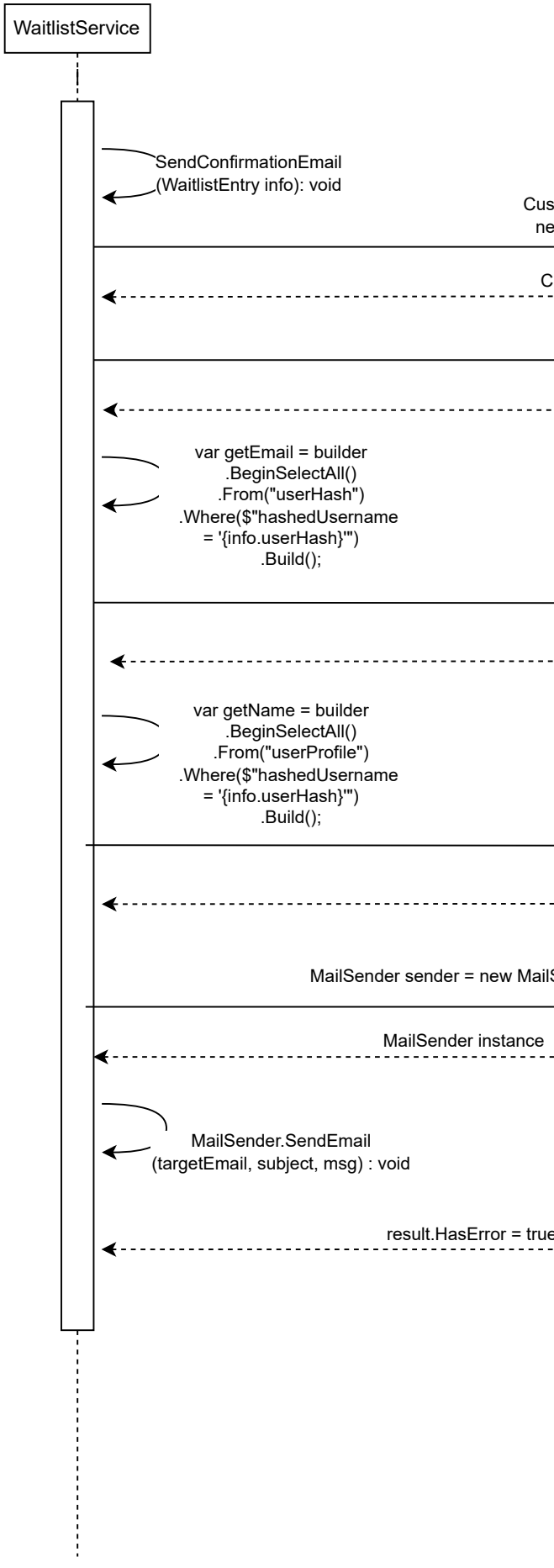


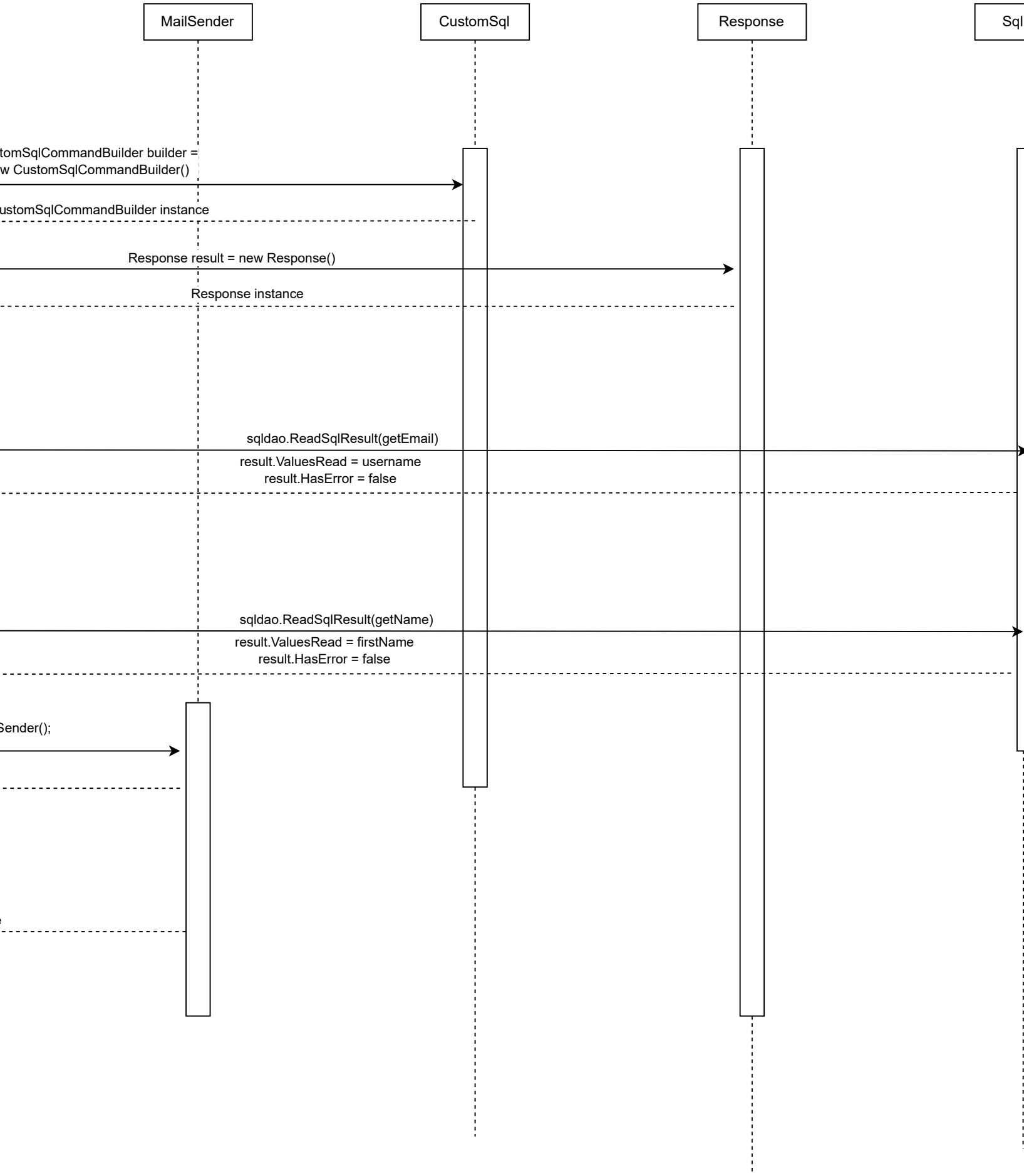

MailSender
CustomSql
Response
Sql
tomSqlCommandBuilder builder =
w CustomSqlCommandBuilder()
ustomSqlCommandBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResult(getEmail)
result.ValuesRead = username
result.HasError = false
sqldao.ReadSqlResult(getName)
result.ValuesRead = firstName
result.HasError = false
Sender();

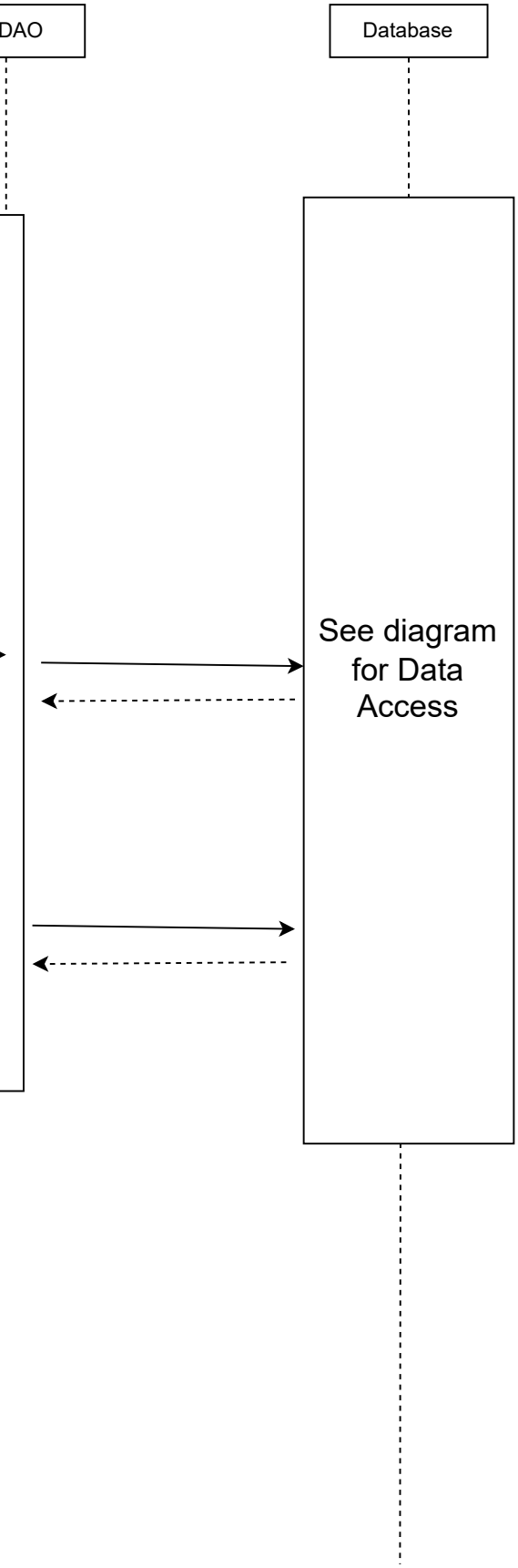

DAO
Database
See diagram for Data Access

Notification Email - Failure

Notification Email - Failure




MailSender
CustomSql
Response
Sql
tomSqlCommandBuilder builder =
w CustomSqlCommandBuilder()
ustomSqlCommandBuilder instance
Response result = new Response()
Response instance
sqldao.ReadSqlResult(getEmail)
result.ValuesRead = username
result.HasError = false
sqldao.ReadSqlResult(getName)
result.ValuesRead = firstName
result.HasError = false
Sender();


DAO
Database
See diagram for Data Access