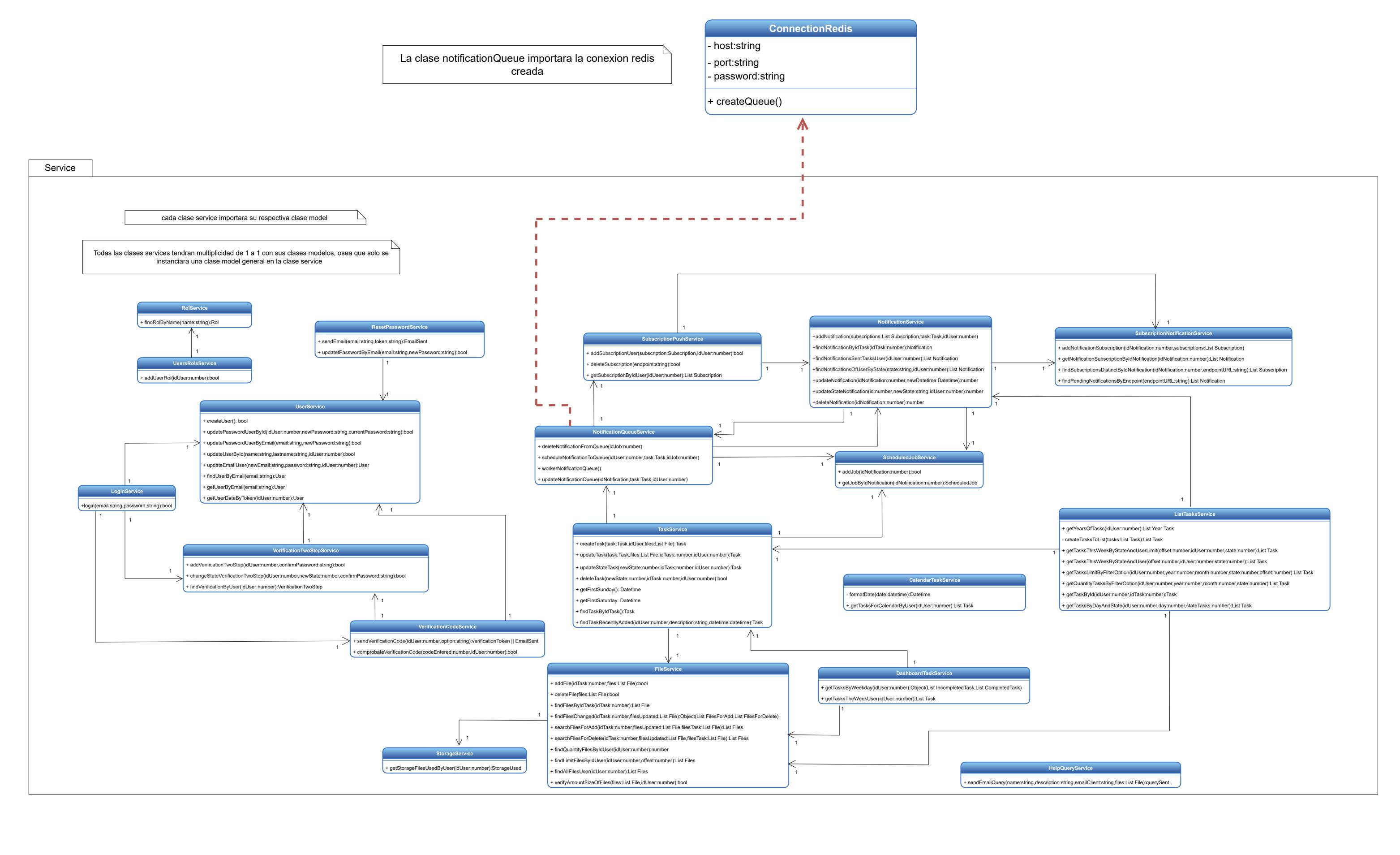


Cada clase model importara la conexion creada

# ConnectionMysql - host:string - user:string - passoword:string - nameDB:string + createConnection()



# Controller

Todas los controladores importaran su clase service correspondiente

### StorageController

+ getStorageFilesUsedByUser(req,res):res.status(httpResponseCode).json(errorObject,Storage)

### UserController

- + createUser(req,res):res.status(httpResponseCode).json(errorObject,bool)
- + getUserByEmail(req,res):res.status(httpResponseCode).json(errorObject,User)
- + getUserDataByToken(req,res):res.status(httpResponseCode).json(errorObject,User)
- + updatePasswordUserById(req,res):res.status(httpResponseCode).json(errorObject, bool)
- + updateUserById(req,res):res.status(httpResponseCode).json(errorObject,bool) + updateEmailUser(req,res):res.status(httpResponseCode).json(errorObject,bool)

loginController

+ login(req,res):res.status(httpResponseCode).json(errorObject,token,bool)

### NotificationController

Todas las operaciones de los controladores dependen de

una autenticación, excepto get User By Email () en

<u>UserController</u>

+ getNotificationsSentTasksUser(req,res):res.status(httpResponseCode).json(errorObject,List Notification) + getNotificationsOfUserByState(req,res):res.status(httpResponseCode).json(errorObject,List Notification)

updateStateNotification(req,res):res.status(httpResponseCode).json(errorObject,number)

# TaskController

- + createTask(req,res):res.status(httpResponseCode).json(errorObject,Task)
- + updateTask(req,res):res.status(httpResponseCode).json(errorObject,Task)
- + updateStateTask(req,res):res.status(httpResponseCode).json(errorObject,Task)
- + delete(req,res):res.status(httpResponseCode).json(errorObject,bool)
- + getYearsOfTasks(req,res):res.status(httpResponseCode).json(errorObject,List Year of Task)
- + getTasksThisWeekByStateAndUserLimit(req,res):res.status(httpResponseCode).json(errorObject,Task)
- + getQuantityTasksThisWeekByStateAndUser(req,res):res.status(httpResponseCode).json(errorObject,number)
- + getTasksByDayAndState(req,res):res.status(httpResponseCode).json(errorObject ,List Task)
- getTasksLimitByFilterOption(req,res):res.status(httpResponseCode).json(errorObject ,List Task)
- + getTasksByFilterOption(req,res):res.status(httpResponseCode).json(errorObject,number)
- + getTaskById(req,res):res.status(httpResponseCode).json(errorObject,Task)

## FileController

+ findQuantityFilesByIdUser(req,res):res.status(httpResponseCode).json(errorObject,number) + findLimitFilesByIdUser(req,res):res.status(httpResponseCode).json(errorObject,List File)

+ sendQueryClient(req,res):res.status(httpResponseCode).json(errorObject,idEmailSent)

### SubscriptionPushController

- + addSubscriptionUser(req,res):res.status(httpResponseCode).json(errorObject,bool)
- + deleteSubscription(req,res):res.status(httpResponseCode).json(errorObject,bool)
- + getSubscriptionByUser(req,res):res.status(httpResponseCode).json(errorObject,List Subscription)

### ResetPasswordController

+ sendEmailToReset(req,res):res.status(httpResponseCode).json(errorObject,EmailSentI)

 $+ \ update Password By Email (req, res): res. status (http Response Code). json (error Object, bool)$ 

# CalendarTaskController

+ getTasksForCalendarByUser(req,res):res.status(httpResponseCode).json(errorObject,List Task)

# DashboardTaskController

+ getTasksByWeekday(req,res):res.status(httpResponseCode).json(errorObject,List Task) + getTasksThisWeekUsers(req,res):res.status(httpResponseCode).json(errorObject,List Task)

## VerificationCodeController

- + sendVerificationCode(req,res):res.status(httpResponseCode).json(errorObject,EmailSent)
- + dataToken(req,res):res.status(httpResponseCode).json(errorObject,Token)
- + comprobateVerificationCode(req,res):res.status(httpResponseCode).json(errorObject,bool)

# VerificatioTwoStepController

- + addVerificationTwoStep(req,res):res.status(httpResponseCode).json(errorObject,bool)
- + getVerificationTwoStepByUser(req,res):res.status(httpResponseCode).json(errorObject,VerificationTwoStep)
- + updateStateVerificationByUser(req,res):res.status(httpResponseCode).json(errorObject,bool)



# Authentication

- + authRequest(req,res):token
- + authRequestByHeader(req,res):token
- + newAccessToken(req,res):token