PointMap

A Geospatial Mapping Tool

Admin Dashboard - User Management: Developer Docs

Document Drafted By:

Team Name	NightWatch
Alfredo Vargas	014270722

April 29, 2019

Table Of Contents

Introduction

Admin Dashboard provides several features to an administrator of this application. For this documentation, the flow and the design of the User Management feature of Admin Dashboard will be documented.

API

RESTful API's following the REST web service architecture.

Server: https://api.pointmap.net/api

Swagger API Documentation

https://app.swaggerhub.com/apis/kfcssodocs/Pointmap/1.0

Technology

Backend

- .NET 4.6.2
- ASP.NET Web API 2
- Entity Framework 6
- MS SQL Server Database

Frontend

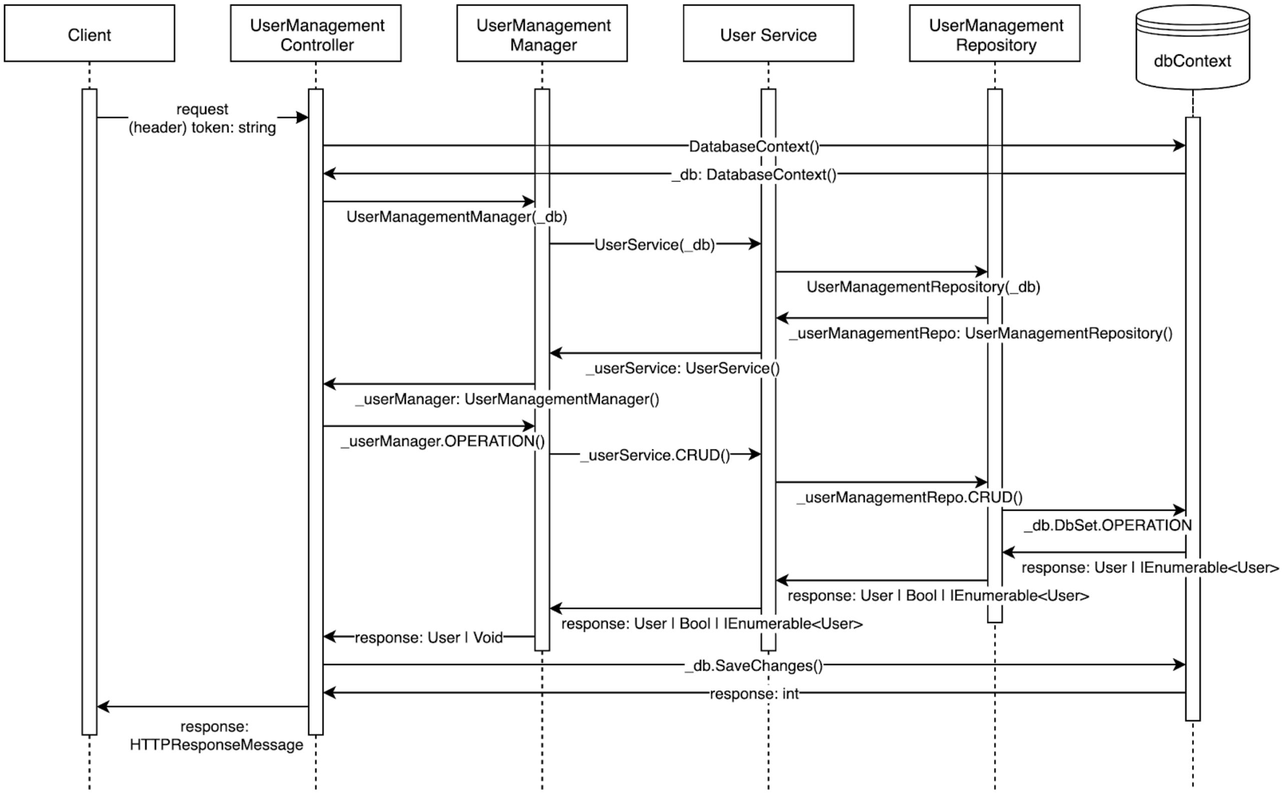
- Vue
- Axios
- Vuetify

Hosting

- AWS EC2 Instance
    - Windows Server 2016
    - Microsoft IIS 10.0
- AWS RDS Instance
    - MS SQL Server
- AWS Route53

Design

Architecture


Client
UserManagement Controller
UserManagement Manager
User Service
UserManagement Repository
dbContext
request
(header) token: string
DatabaseContext()
_db: DatabaseContext()
UserManagementManager(_db)
UserService(_db)
UserManagementRepository(_db)
_userManagementRepo: UserManagementRepository()
_userService: UserService()
_userManager: UserManagementManager()
_userManager.OPERATION()
_userService.CRUD()
_userManagementRepo.CRUD()
_db.DbSet.OPERATION
response: User | IEnumerable<User>
response: User | Bool | IEnumerable<User>
response: User | Bool | IEnumerable<User>
response: User | Void
_db.SaveChanges()
response: int
response:
HTTPResponseMessage

[1]

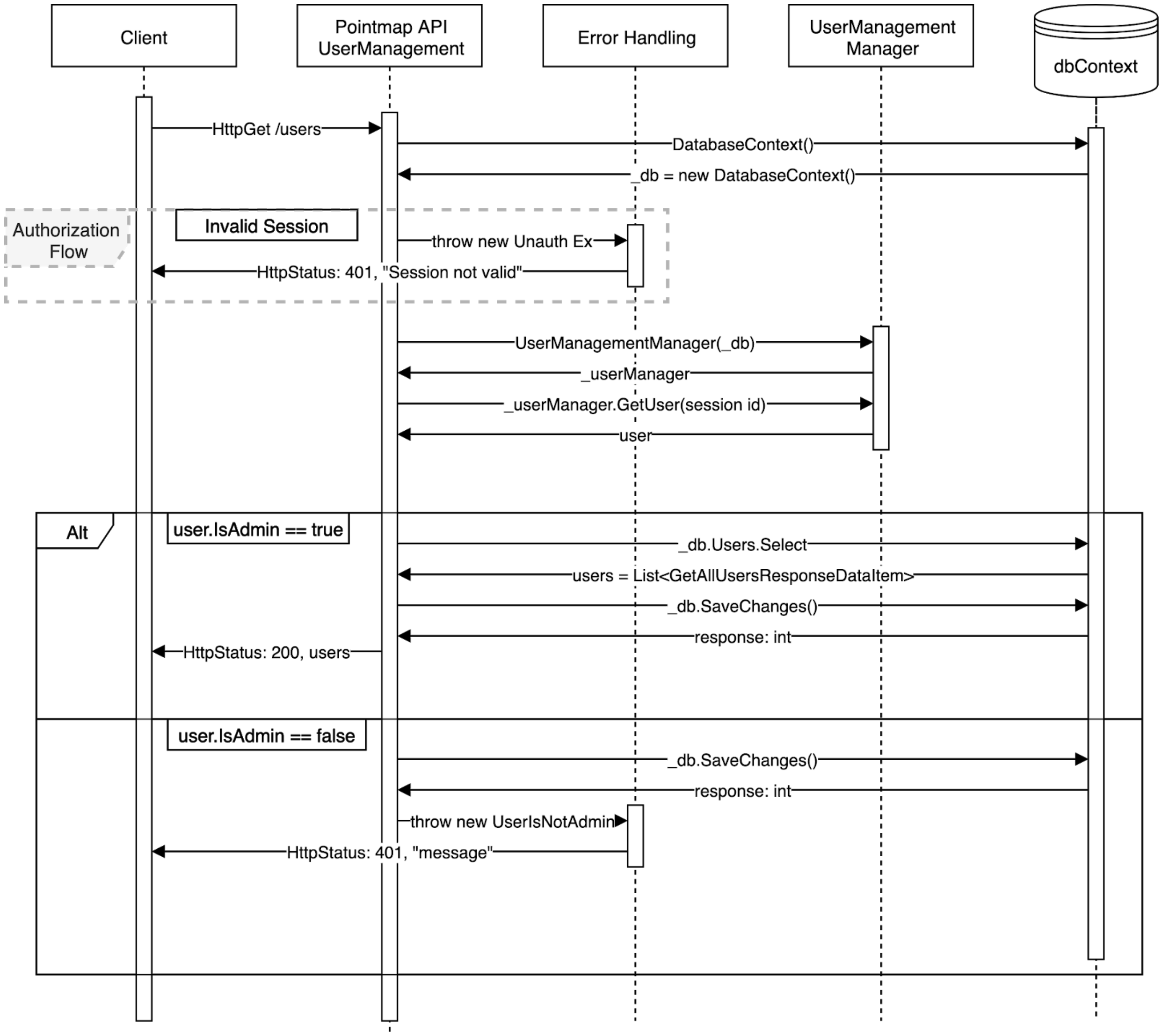
[1] Authorization is not accounted for in the diagram.

Operations

Get All Users

Endpoint: /users

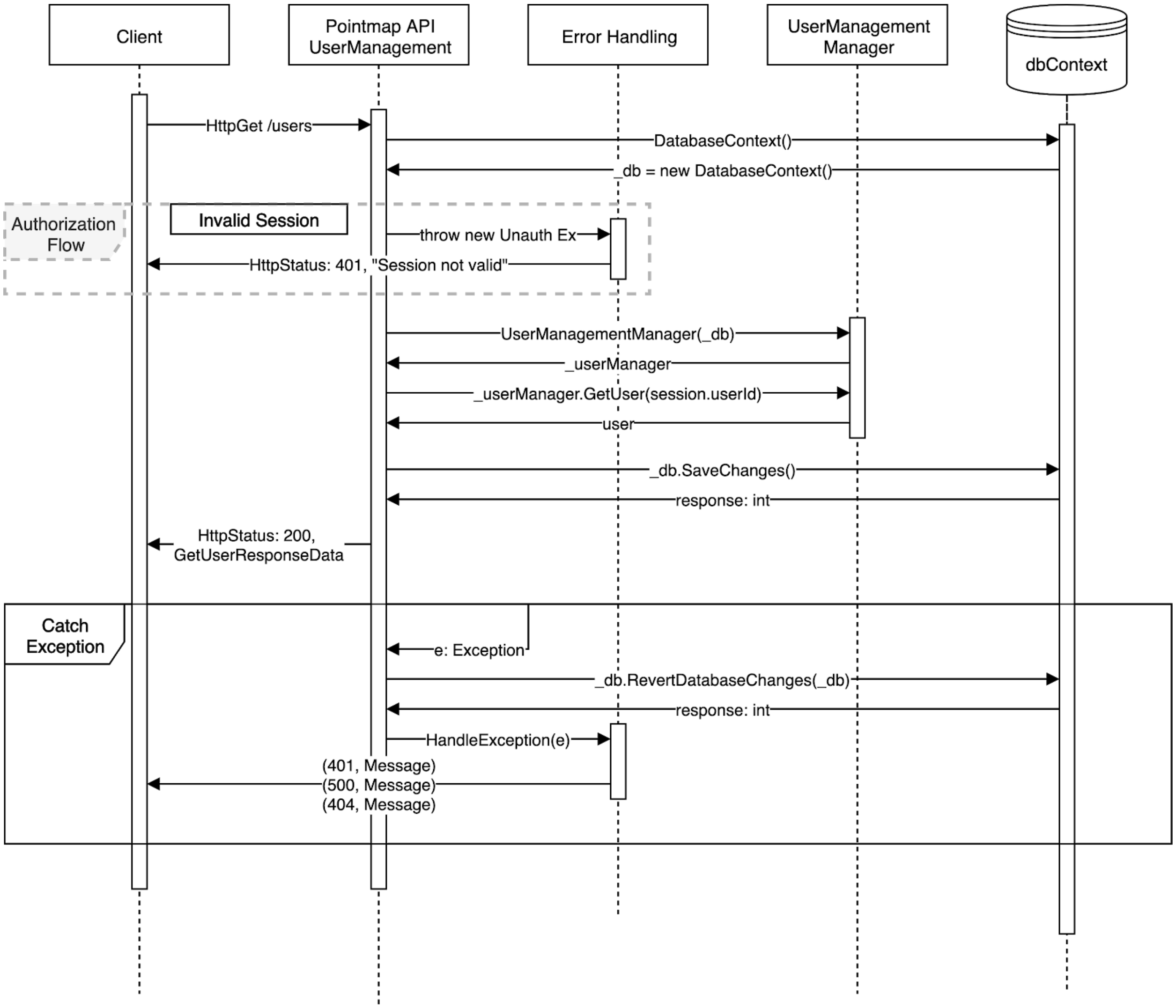
This is the flow of the **GET** request to get all users of the application. The **GET** route will be pinged from the client and will pass in the **token** as a header.



Get a user

Endpoint: /user

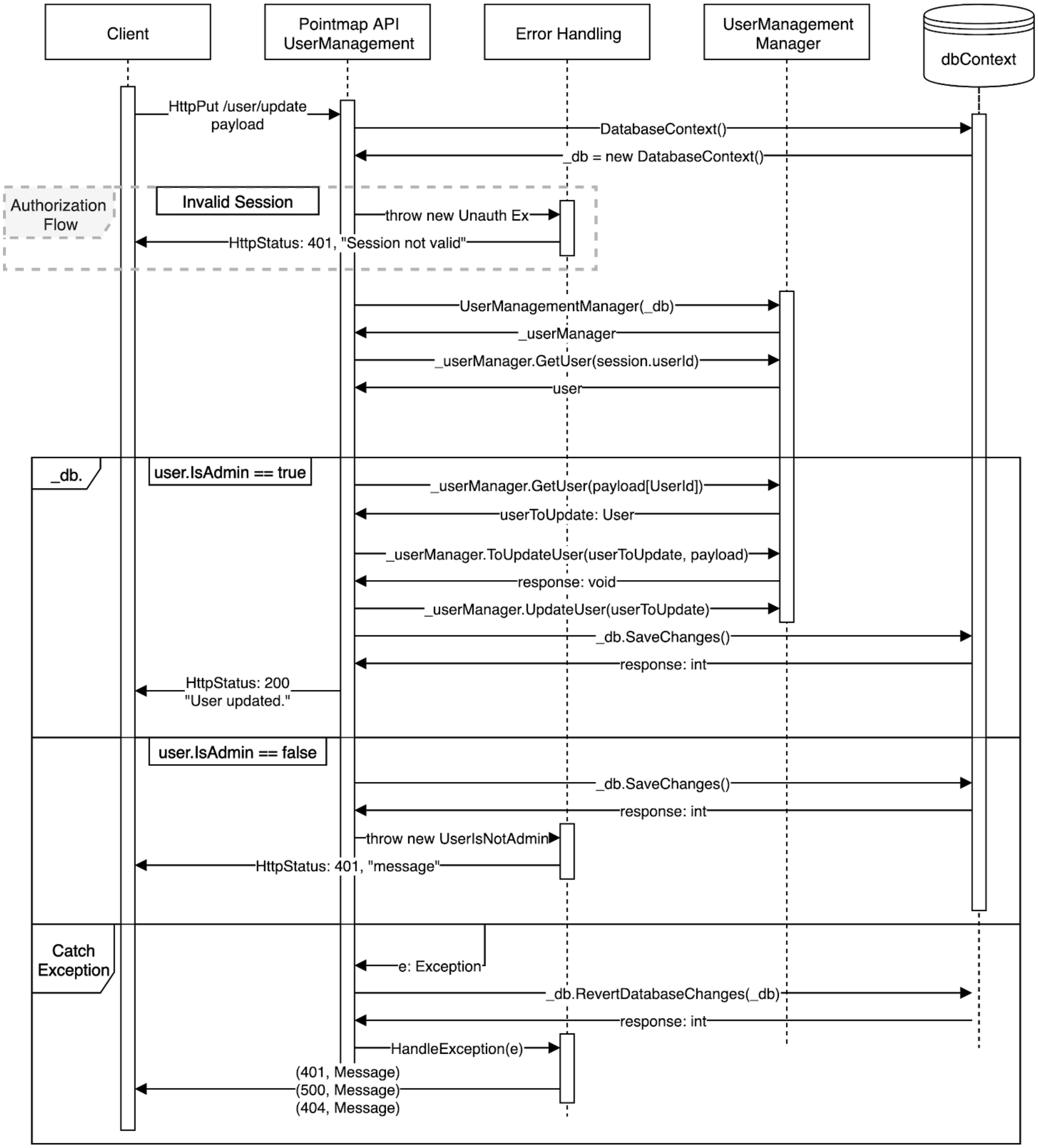
This is the flow of the **GET** request to get a user of the application. The **GET** route will be pinged from the client and will pass in the **token** as a header.



Update a user

Endpoint: /user/update

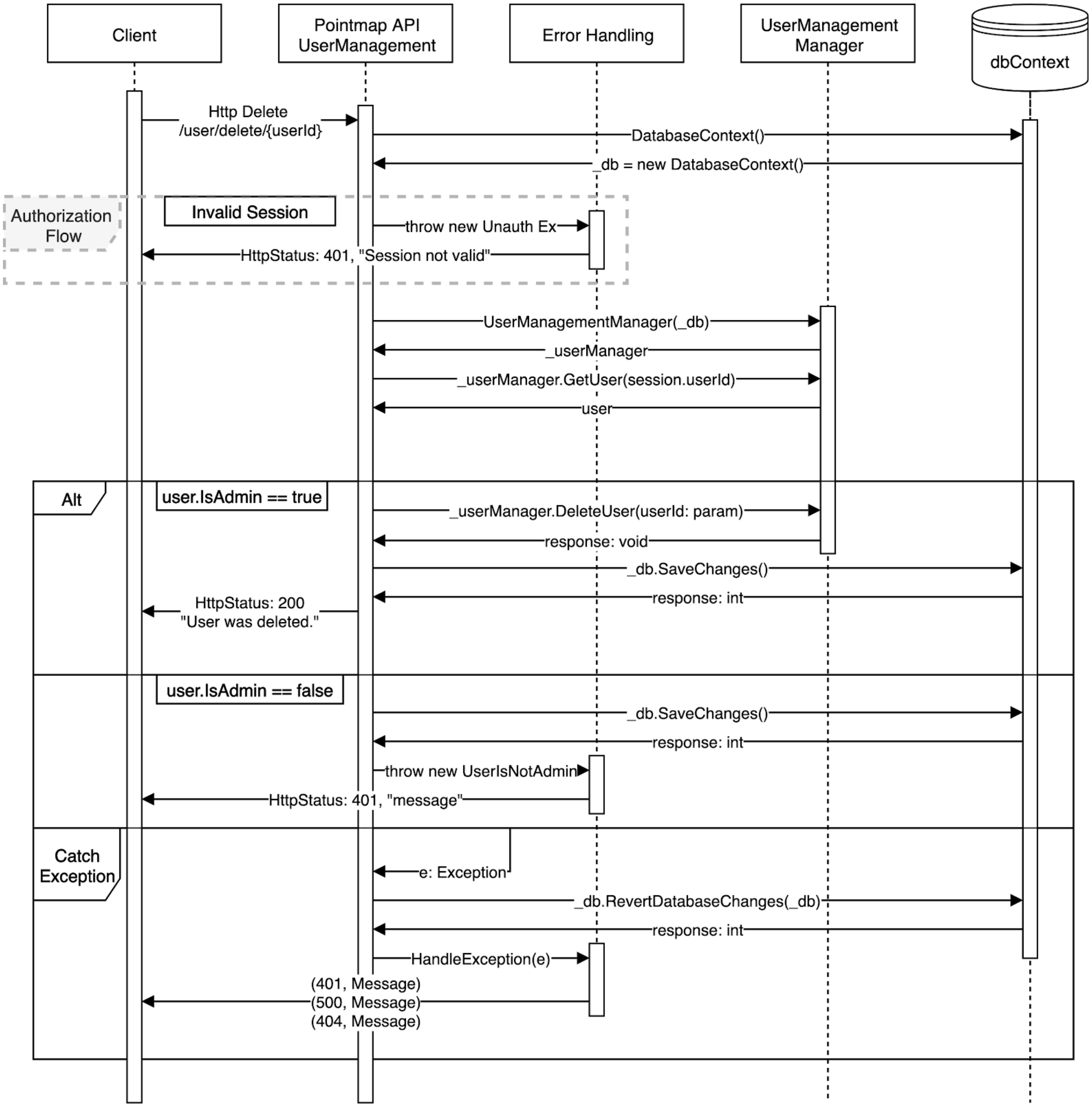
This is the flow of the **PUT** request to update a user of the application. The **PUT** route will be pinged from the client and will pass in the **token** as a header, only a user with administrative privileges will be able to perform the operation.



Delete a user

Endpoint: /user/delete/{userId}

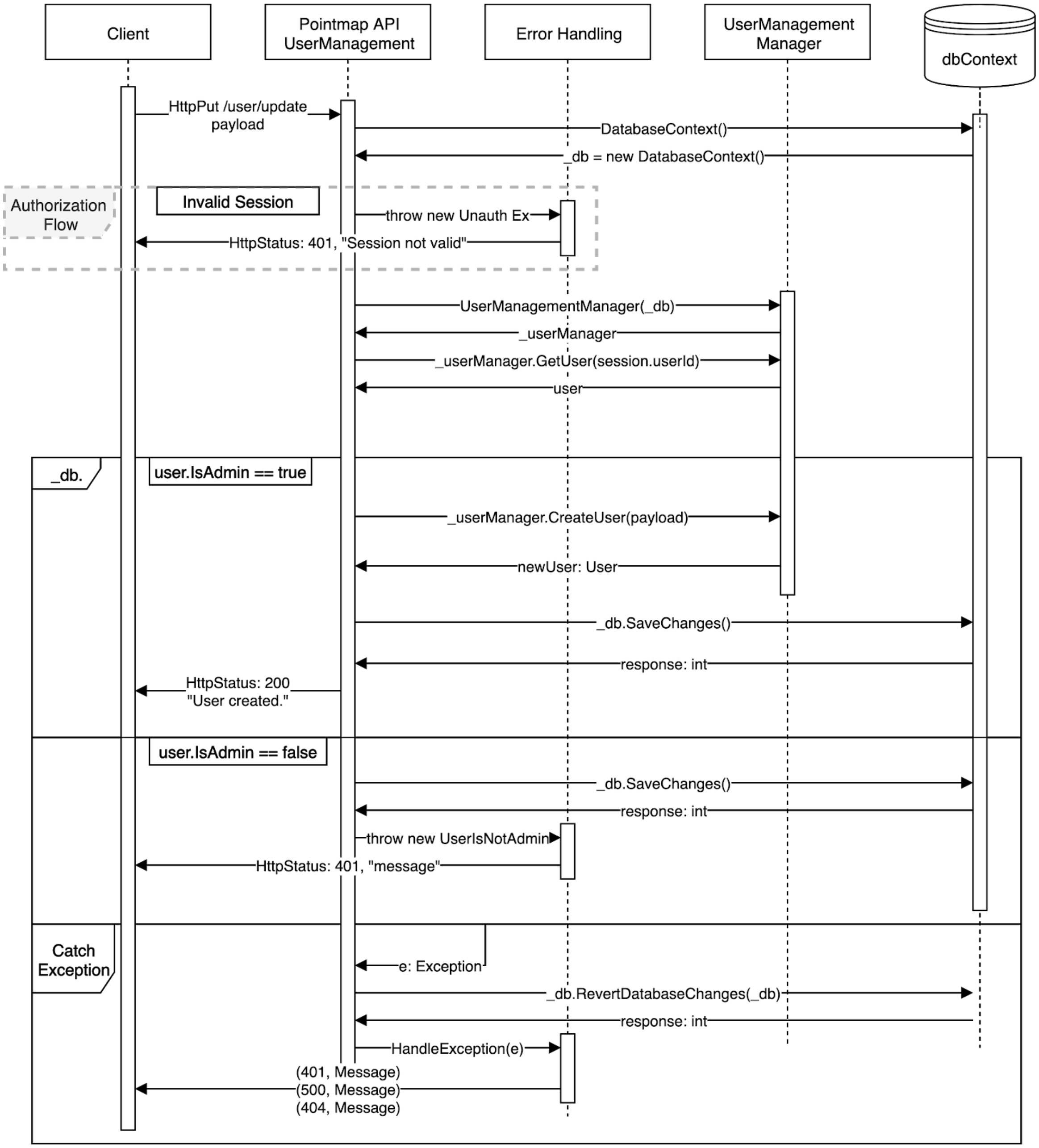
This is the flow of the **DELETE** request to update a user of the application. The **DELETE** route will be pinged from the client and will pass in the **token** as a header, only a user with administrative privileges will be able to perform the operation.



Create a user

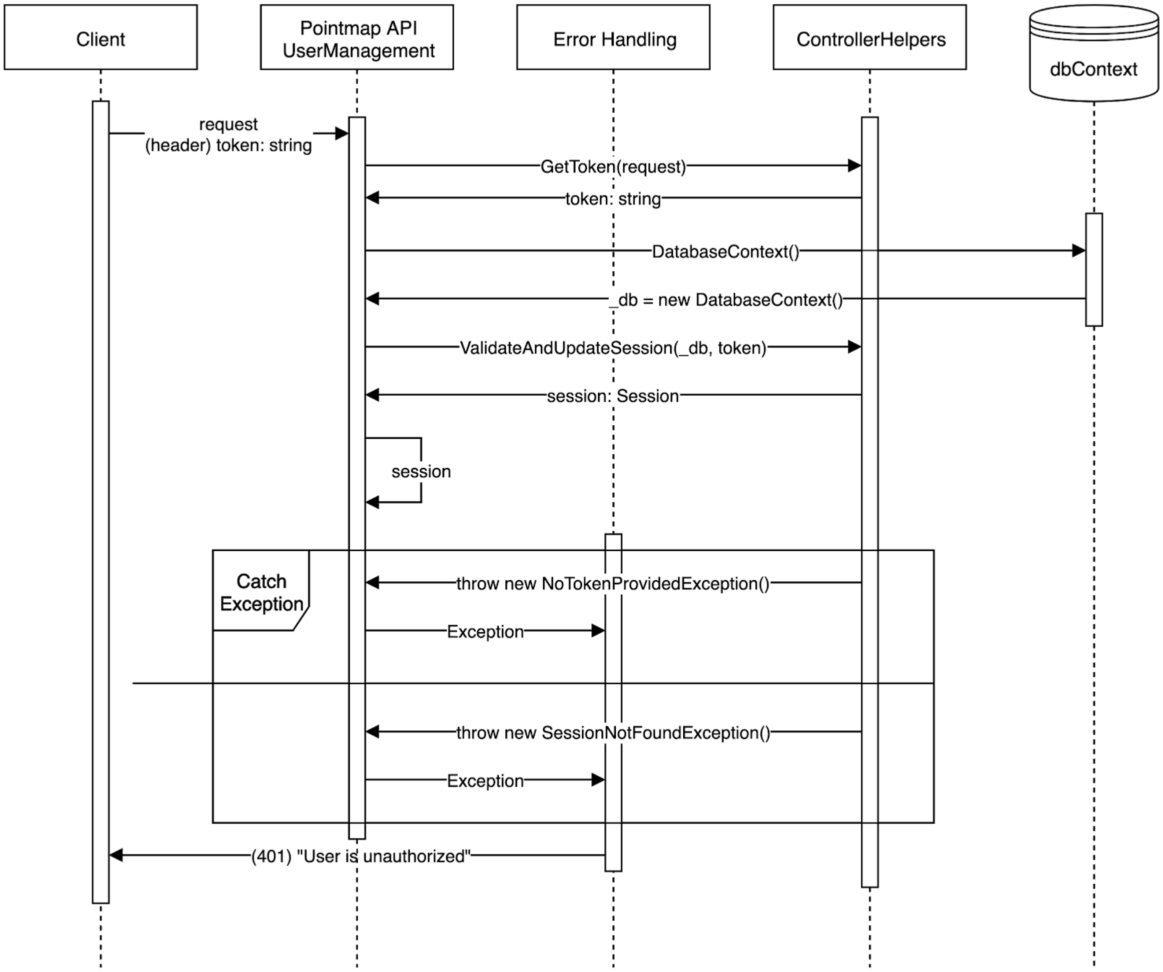
Endpoint: /user/create

This is the flow of the **POST** request to update a user of the application. The **POST** route will be pinged from the client and will pass in the **token** as a header, only a user with administrative privileges will be able to perform the operation.



Authorization

When a route is pinged, all request must contain a **token** header which will contain a string token of the users session. This token will be used to validate the user's session and to identify the user submitting the request. If the token has expired or is invalid, the route will return an unauthorized status code with a message.


Client
Pointmap API UserManagement
Error Handling
ControllerHelpers
dbContext
request
(header) token: string
GetToken(request)
token: string
DatabaseContext()
_db = new DatabaseContext()
ValidateAndUpdateSession(_db, token)
session: Session
session
Catch Exception
throw new NoTokenProvidedException()
Exception
throw new SessionNotFoundException()
Exception
(401) "User is unauthorized"

Testing

Technology: MSTest

Unit Tests

Below are the test methods that are used for unit testing of the user services.

- Create_User_Success
- Create_User_RetrieveNew_Success
- Create_User_Fail_ExceptionThrown
- Create_User_Using_Manager
- Create_User_Using_Manager_NotRealEmail
- Delete_User_Success
- Delete_User_NonExisting
- Update_User_Success
- Update_User_NonExisting_why
- Update_User_OnRequiredValue
- Get_User_Success
- Get_User_NonExisting
- Disable_User_Success
- Enable_User_Success
- Toggle_User_Success
- Get_All_Users_Success

Integration Tests

Below are the test methods that are used for integration testing of the UserManagement Controller in Pointmap Web API

- NoSessionToken_Unauthorized_401
- FakeSessionToken_Unauthorized_401
- GetAllUsers_Authorized_200
- GetAllUsers_Unauthorized_401
- GetUser_Authorized_200
- DeleteUser_Authorized_200
- DeleteUser_Unauthorized_401
- DeleteNonExistingUser_Authorized_404
- UpdateUser_Authorized_200
- UpdateUser_Unauthorized_401
- UpdateNonExistingUser_Authorized_404
- UpdateUser_InvalidModelPayload_400
- UpdateUser_NonexistingManager_404
- UpdateUser_InvalidManagerGuid_400
- CreateNewUser_Authorized_201
- CreateNewUser_Unauthorized_401
- CreateNewUser_InvalidUsername_400
- CreateNewUser_InvalidManagerGuid_400
- CreateNewUser_NonexistingManager_404

E2E Testing

Technology

- puppeteer 1.13.0
- mocha 6.1.4
- chai 4.2.0