BerghAdmin - Documentation

table of contents

Generated on: 2023-10-16 09:54:37 +0200

1. Introduction

2. README

- get the code at https://github.com/florisz/BerghAdmin/settings/access
- SQL Server
  - install a SQL server on your local (or antoher accessible) machine
  - create a database with name 'BIHZ2021' with a user/password with read/write permissions (can actually be any name but this one is most convenient)
  - install latest entity framework client tools; see https://docs.microsoft.com/en-us/ef/core/cli/dotnet
    just calling 'dotnet tool install --global dotnet-ef' will do the job
- Secrets
  - get secrets file from another developer (is not in the repo)
  - store secrets file in solution directory (keep calm, git will not synchronize it to other repos)
  - load secrets into your private secret store with ./SetSecrets.ps1
  - change the "DatabaseConfiguration:ConnectionString" with your own specific value (use the values you've chosen above and one of the commands below)
  - run 'dotnet ef database update' from the command line (this should create an empty database)
- Source code
  - start Visual studio and load the solution file
  - Open the 'Test Explorer' window and run all unit tests (this should give you a safe feeling)
  - debug/run the default project (should be BerghAdmin)

Help with setting secrets in local store:

```
dotnet user-secrets clear --project ./BerghAdmin
dotnet user-secrets list --project ./BerghAdmin
dotnet user-secrets remove "sectie:setting" --project ./BerghAdmin
dotnet user-secrets add "sectie:setting" --project ./BerghAdmin
```

3. Domain overview

3.1. Business background

BerghAdmin will be used by the backoffice people of the charity organization Bergh in het Zadel (BIHZ). With the aid of many donors BIHZ collects money to fight cancer or as the slogan says: "to buy time for cancer patients".

BIHZ is currently active in:

- maintaining a group of Ambassadors which donate every year
- organizing a cycle tour every four year to raise money
- organizing multiple golfevents every year also to raise money

Next to this money is raised to ad-hoc events and people and organizations donate money occasionaly.

To accomodate this (automated) support is needed for all administrative roles. In the next paragraph the most common tasks are mentioned. The BerghAdmin application is meant to deliver this support to all volunteers in administrative roles. Focused at freeing up their time for other more fun stuff, achieving one central consistent administration and conform to legislation (financially and AVG proof).

3.2. Context

Functionally the domain for BerghAdmin is:

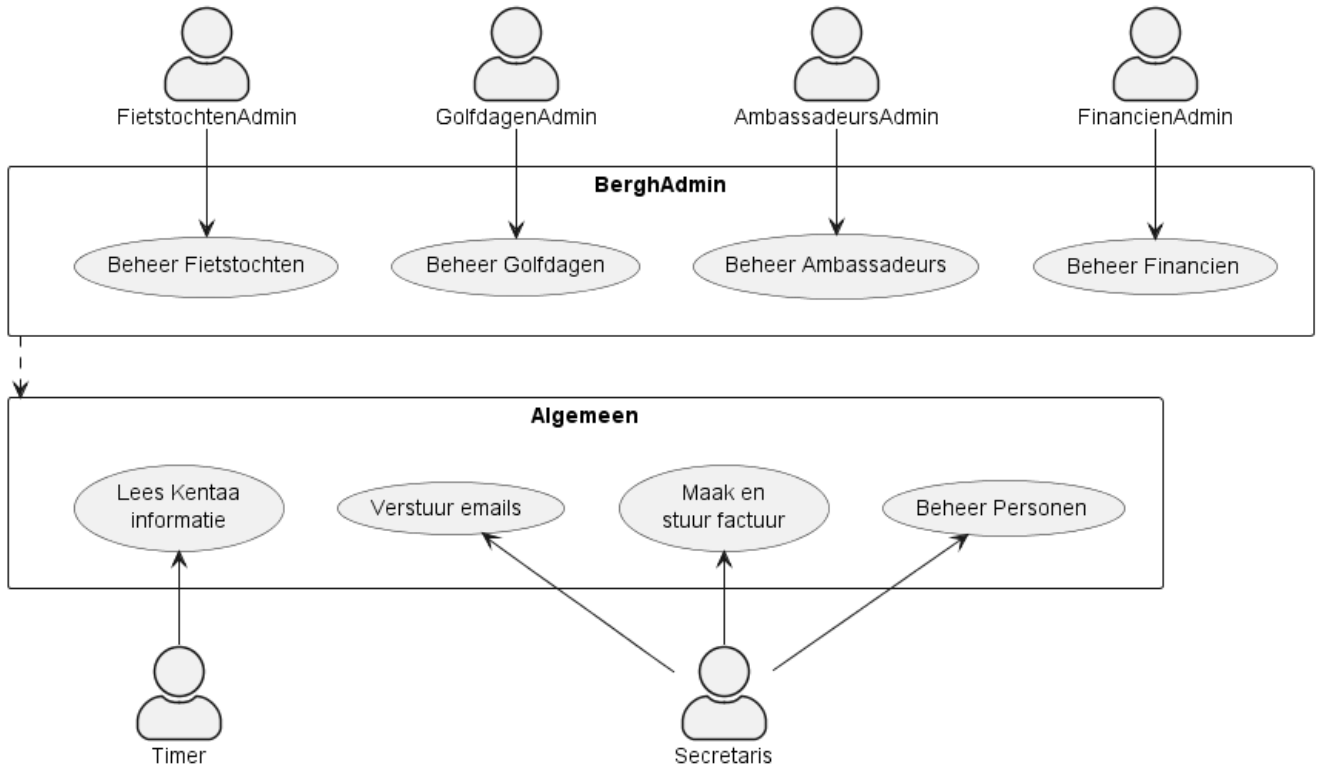

Figure 1. Context diagram BerghAdmin

For each of the four subdomains (Ambassadeurs, Fietstochten, Golfdagen and Fiancien) specific support is needed to manage the subdomain. However there is generic support in: creating/sending emails, creating/sending invoices and the (central) administration of all people connected to the BIHZ-organization.

For the handling (administration, payment and invoicing) of mainly small gifts BIHZ relies on the Kentaa service offering. With Kentaa, donors can define their own donate page thru which money is raised for BIHZ. On regular intervals (a Timer) the BerghAdmin system will read the Kenta

information and link it to the appropriate persons in the administration. This will enable a smooth financial management of all incoming gifts.

4. Analysis

Functionalities

- central donateurs mgmt
- fietstochten
- connection Kentaa
- golfdagen
- facturen (ambassadeurs)
- email sending
- history

5. Design

5.1. Structure

The high level class model for the application:

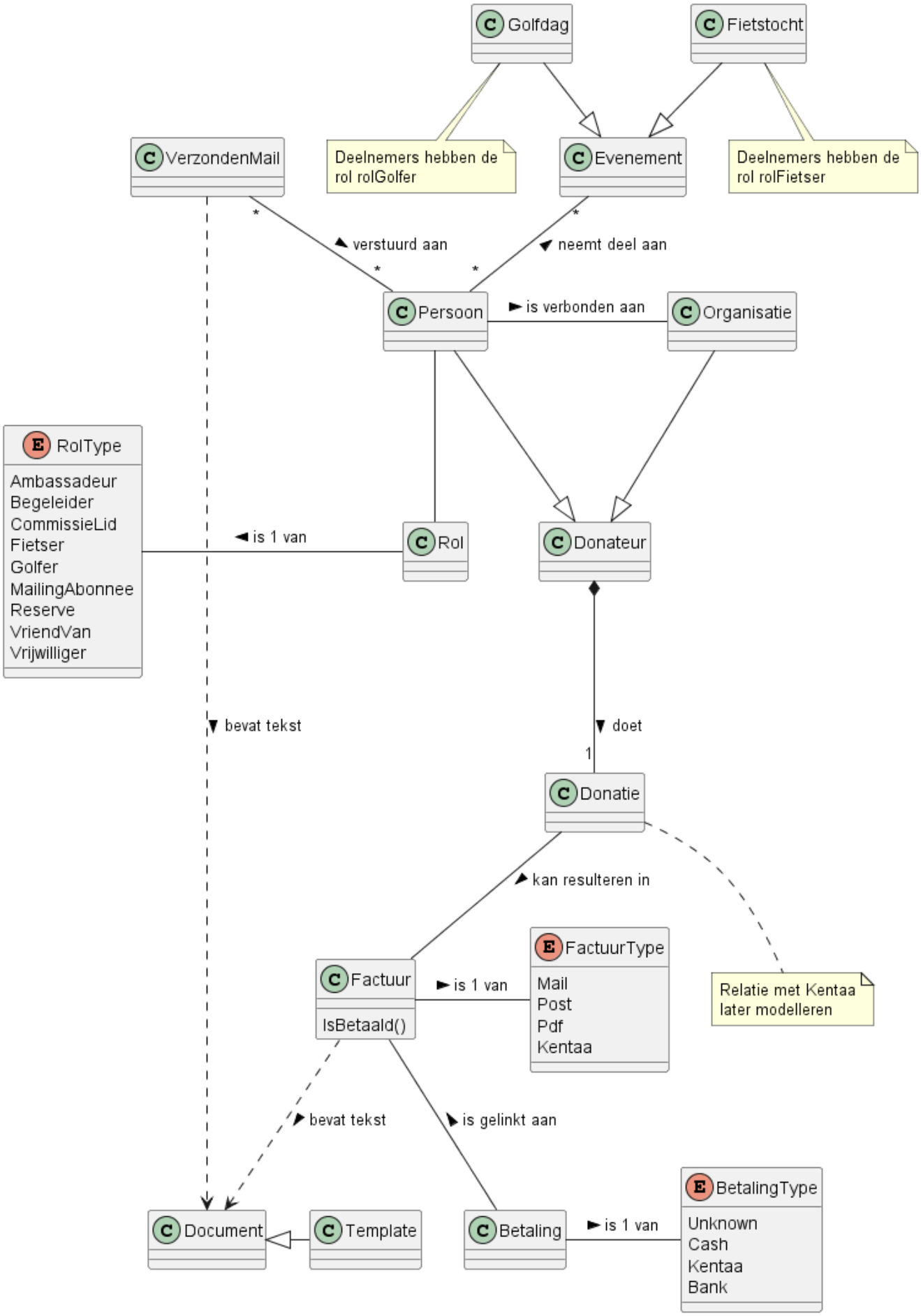

Figure 2. High level class model BerghAdmin

Kentaa is a separate organization which is handling gifts and collecting money from a large group

of donateurs. Mainly for the money earned by cyclists but also other 'projects' are handled with the aid of this service. Kentaa stores its information in the following structure:

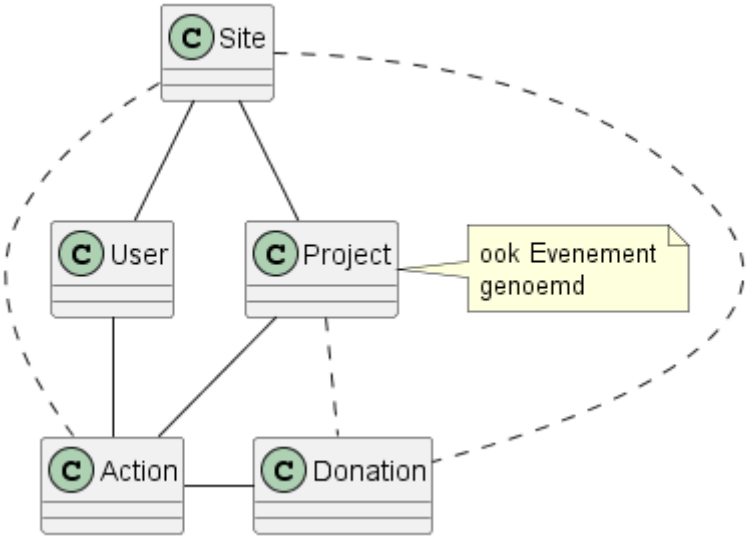

Figure 3. Data model Kentaa

5.2. Scenario’s

> Look at the Integration test 'FullKentaaIntegrationTest' in the BerghAdmin.Tests project for an overview of the described mechanism and dependencies.

Design of the scenario how to read in the Kenta information. As can be seen from the class model Kentaa stores its information in 5 seperate entities (classes). The Site class in Kentaa corresponds to BIHZ and is therefore irrelevant to BerghAdmin. The following table shows the correspondence between the Kentaa and the BerghAdmin classes:

Kentaa class	**Bihz class**	**Data is linked to**
User	BihzUser	Persoon, based on email address stored in BihzUser
Project	BihzProject	Evenement, based on title stored in BihzProject
Action	BihzAction	Persoon, based on email address stored in BihzAction
Donation	BihzDonatie	Step 1: link to Persoon via the ActionId in the BihzDonatie Step2: add/update the Donatie via the BihzDonatie.Id

The Kentaa class is regularly retrieved from the Kentaa API by an Azure funtion and mapped to the Bihz class. The content of the Bihz class is sent to one of the 4 corresponding endpoints of the BerghAdmin web application. When received the incoming is processed as described in the table above.

This illustrated with:

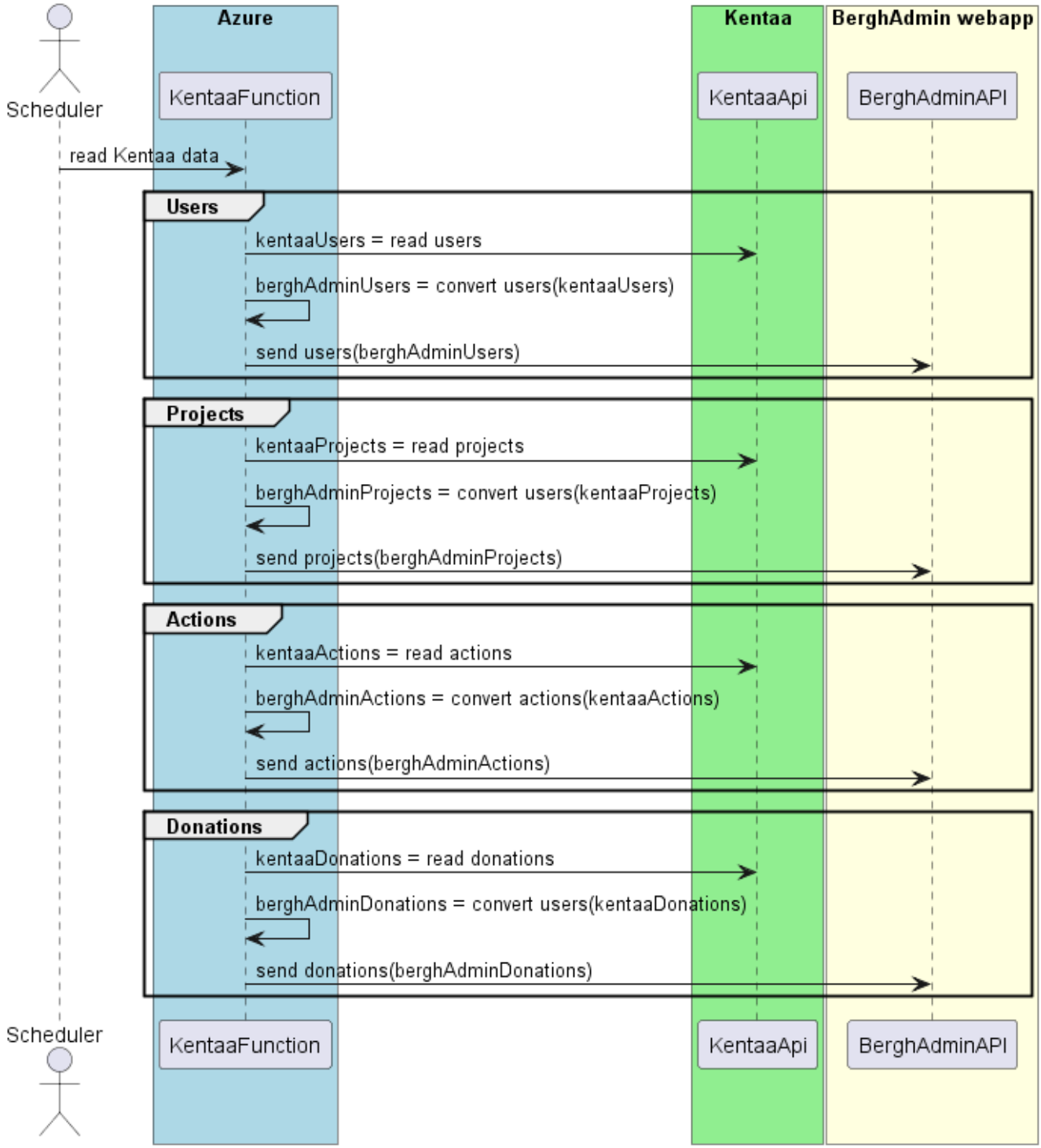

Figure 4. Kentaa Api read sequence

To handle all incoming data in the BerghAdmin Api, four endpoints have been created. one for each of the data types received from the Kentaa function. The following paragraphs will show this

5.2.1. Projects

Any incoming project must be mapped to an evenement. It is required the title of the incoming project must match the (unique) title of the evenement. This is shown in:

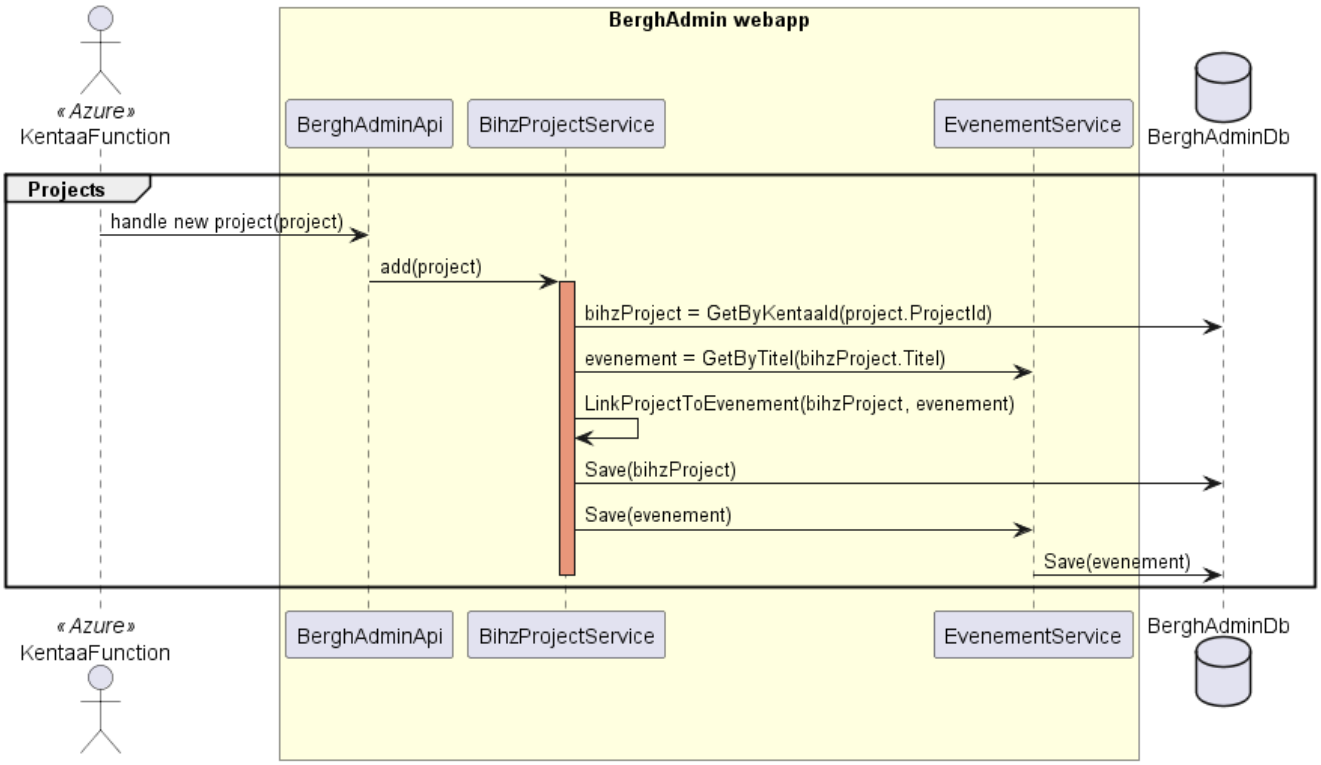

Figure 5. Handle incoming Kentaa Projects sequence

5.2.2. Acties

An actie in Kentaa corresponds to the webpage of a user. With this page the user is collecting money for Bergh in het Zadel. Any incoming actie must therefore be mapped to a persoon. It is required the email address of the incoming actie must match the (unique primary) email address of the persoon. This is shown in:

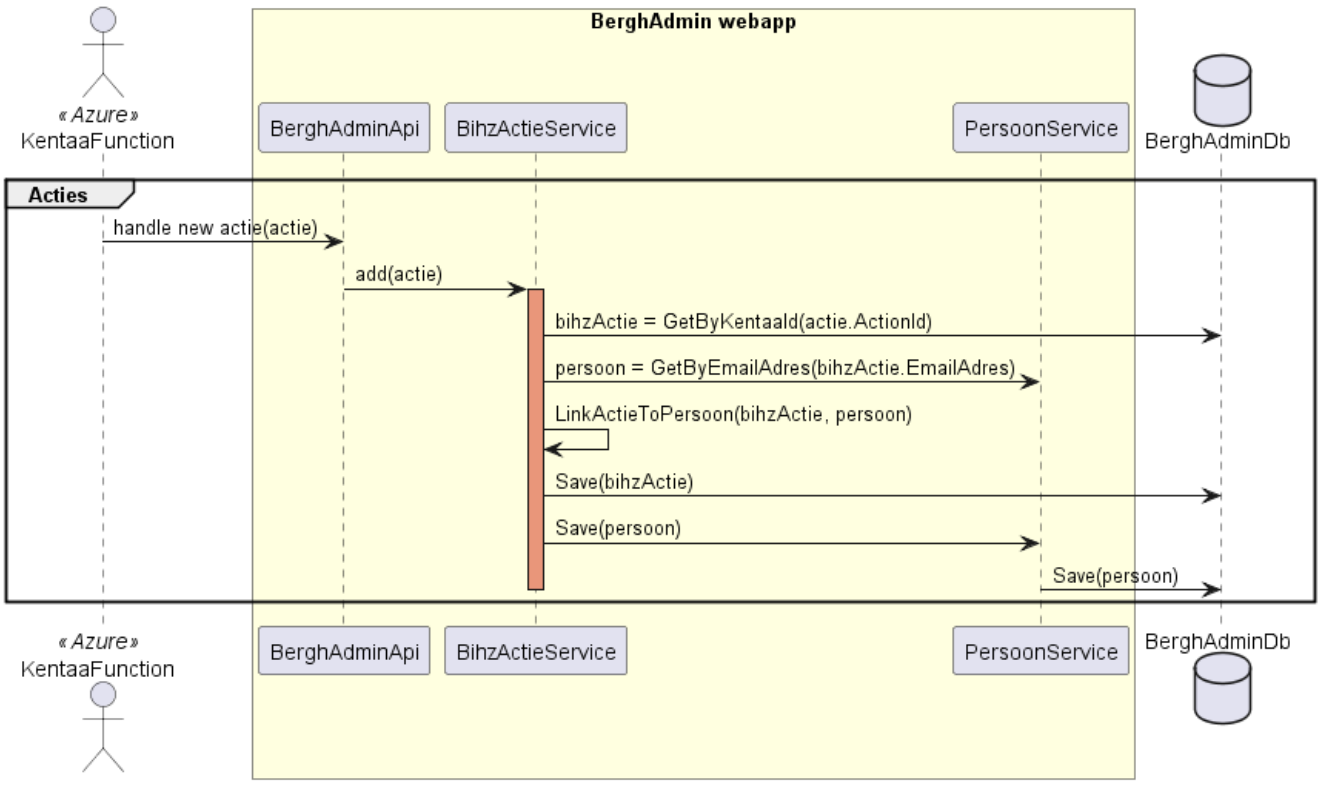

Figure 6. Handle incoming Kentaa Acties sequence

5.2.3. Users

Incoming users are treated pretty similar as Acites (see previous paragraph). This is because each user is linked to a Persoon also. Any incoming user must be mapped to a persoon. It is required the email address of the incoming user must match the (unique primary) email address of the persoon. This is shown in:

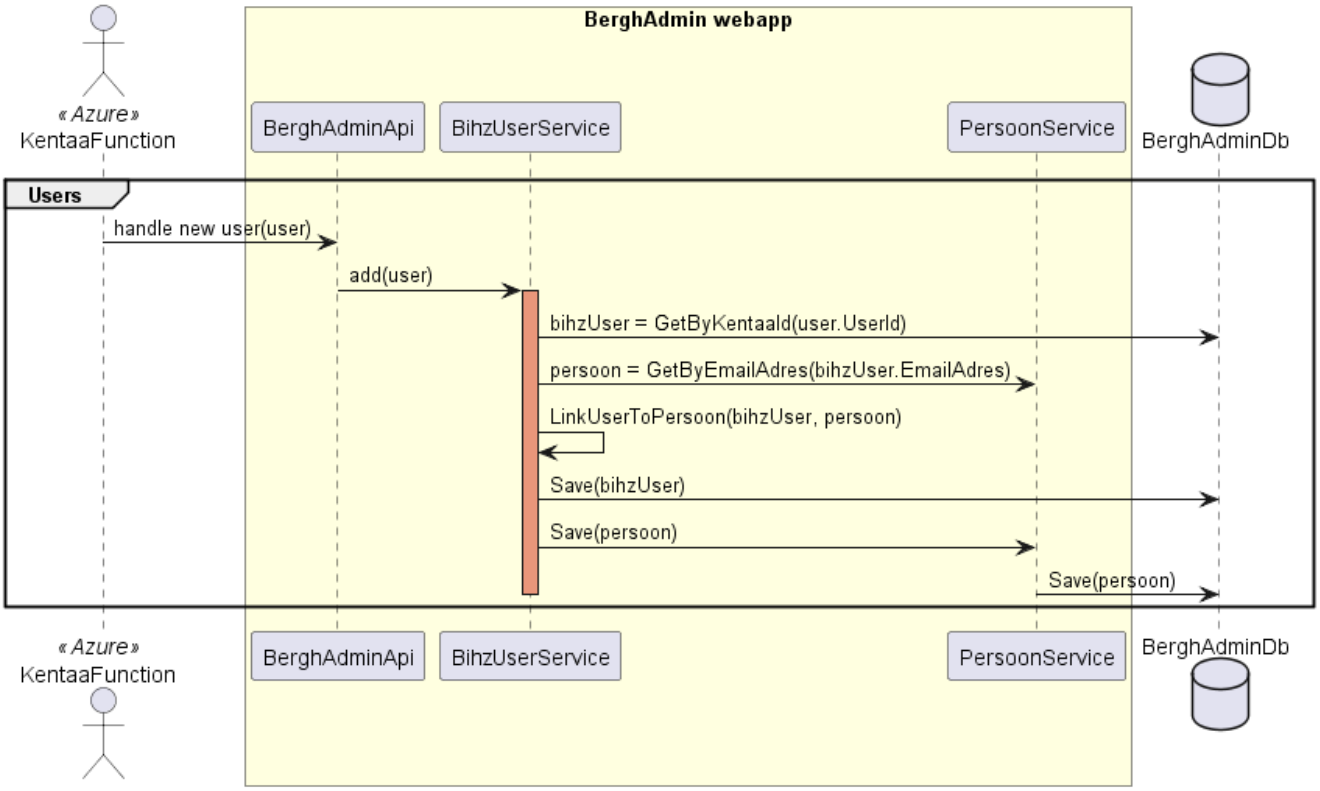

Figure 7. Handle incoming Kentaa Users sequence

5.2.4. Donaties

Incoming donaties are treated slightly different than the previous three types. This is mainly because donaties in Kentaa are linked to acties while each donatie in the BIHZ domain must be linked to a persoon. This results in the following design:

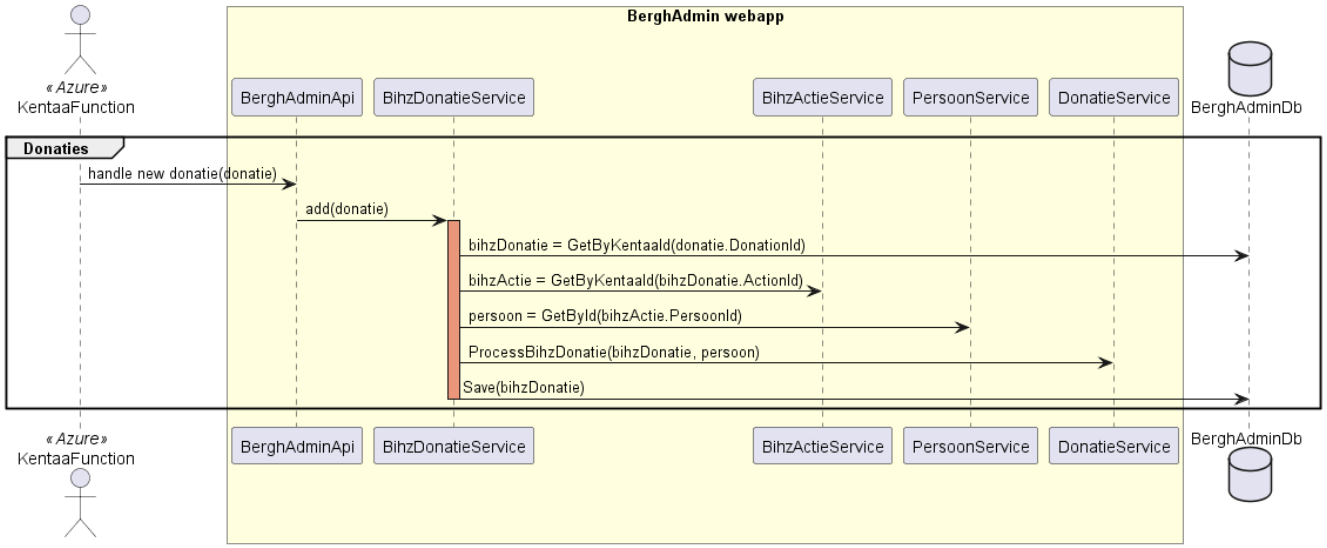

Figure 8. Handle incoming Kentaa Donaties sequence

6. Infrastructure

picture with:

- webapp @Digital Ocean
- Kentaa
- Rabobank (CSV downloads)
- SQL Server MijnServer
- Azure function
- Mailjet
- Syncfusion

== Linux installation

future?

- Nginx: https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-in-ubuntu-16-04
- Host ASP.NET Core on Linux with Nginx https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-6.0
- Host and deploy Blazor Server https://docs.microsoft.com/en-us/aspnet/core/blazor/host-and-deploy/server?view=aspnetcore-6.0#linux-with-nginx
- ASP.NET Core SignalR hosting and scaling https://docs.microsoft.com/en-us/aspnet/core/signalr/scale?view=aspnetcore-6.0#linux-with-nginx