Sights - Tourist Attractions

Gruppnamn och gruppmedlemmar

Annika Sörenstam - Jenny Dyews, Louise Littecke, Fredric Lundgren, Teodor Fredriksson, Alexandra Salén Sammueang, Maria Kilsved och Frans Nilsson Lidström

Repository

https://github.com/MariaKilsved/sights.git

Hur man bygger och startar

Bygg backend med VisualStudio. Kör frontend med Live Server i VisualStudio Code, öppna mappen sights-frontend eller använd http-server i terminalen för https. Backend kan testas separat från frontend genom Swagger.

För att få god översikt över databasen rekommenderar vi DB Browser for SQLite:

https://sqlitebrowser.org/

HTTPS

För att starta frontend med HTTPS behöver man installera http-server med npm:

```
npm install http-server -g
```

Installation med scoop (Windows):
Om du inte har scoop ladda ner här:

https://scoop.sh/

Eller skriv följande i PowerShell:
```
irm get.scoop.sh | iex
```

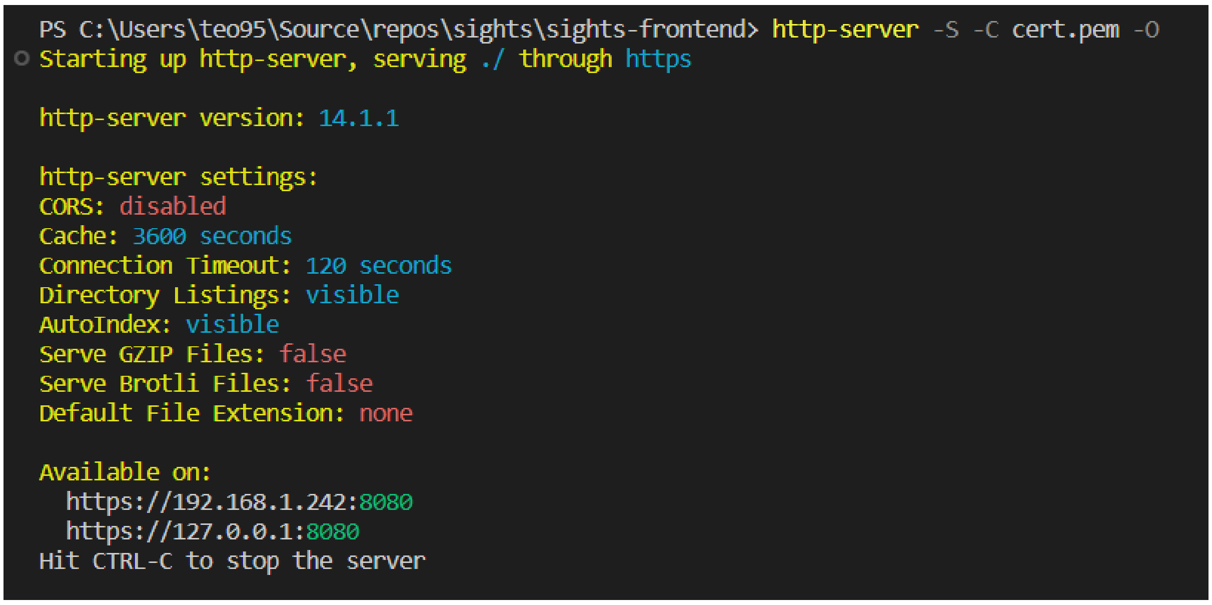
Ställ dig sedan i sights-frontend mappen i terminalen och skriv in följande för att skapa ditt certifikat:
```
scoop bucket add extras
scoop install mkcert
mkcert -install
mkcert localhost
```

Döp om filerna `localhost-key.pem` till `key.pem` och `localhost.pem` till `cert.pem`.
För att starta servern skriv även följande i sights-frontend mappen:

```
http-server -S -C cert.pem -o
```

Efter att ha startat servern i Visual Studio kan du nu skriva in https://localhost:8080/ i din webbläsare. Du kan behöva byta ut 8080 i din url, baserat på vilken port din https server körs på. Du kan se i din terminal vilken port som används när du skriver in:

```
http-server -S -C cert.pem -o
```



```
PS C:\Users\teo95\Source\repos\sights\sights-frontend> http-server -S -C cert.pem -O
Starting up http-server, serving ./ through https

http-server version: 14.1.1

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  https://192.168.1.242:8080
  https://127.0.0.1:8080
Hit CTRL-C to stop the server
```

Installation med brew (Mac/Linux):

Först behöver man installera OpenSSL.

Officiell nedladdningslänk:

https://www.openssl.org/source/

Alternativt kan man installera med brew i terminalen:

```
brew install openssl
```

Ställ dig sedan i sights-frontend mappen i terminalen och skriv in följande för att skapa ditt certifikat:

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout
key.pem -out cert.pem
```

Två nya filer bör då ha dykt upp i mappen sights-frontend. Döp om filen `localhost-key.pem` till `key.pem` och `localhost.pem` till `cert.pem`.

För att starta servern skriv även följande i sights-frontend mappen:

```
http-server -S -C cert.pem -o
```

User secrets

Lägg till följande user secrets:

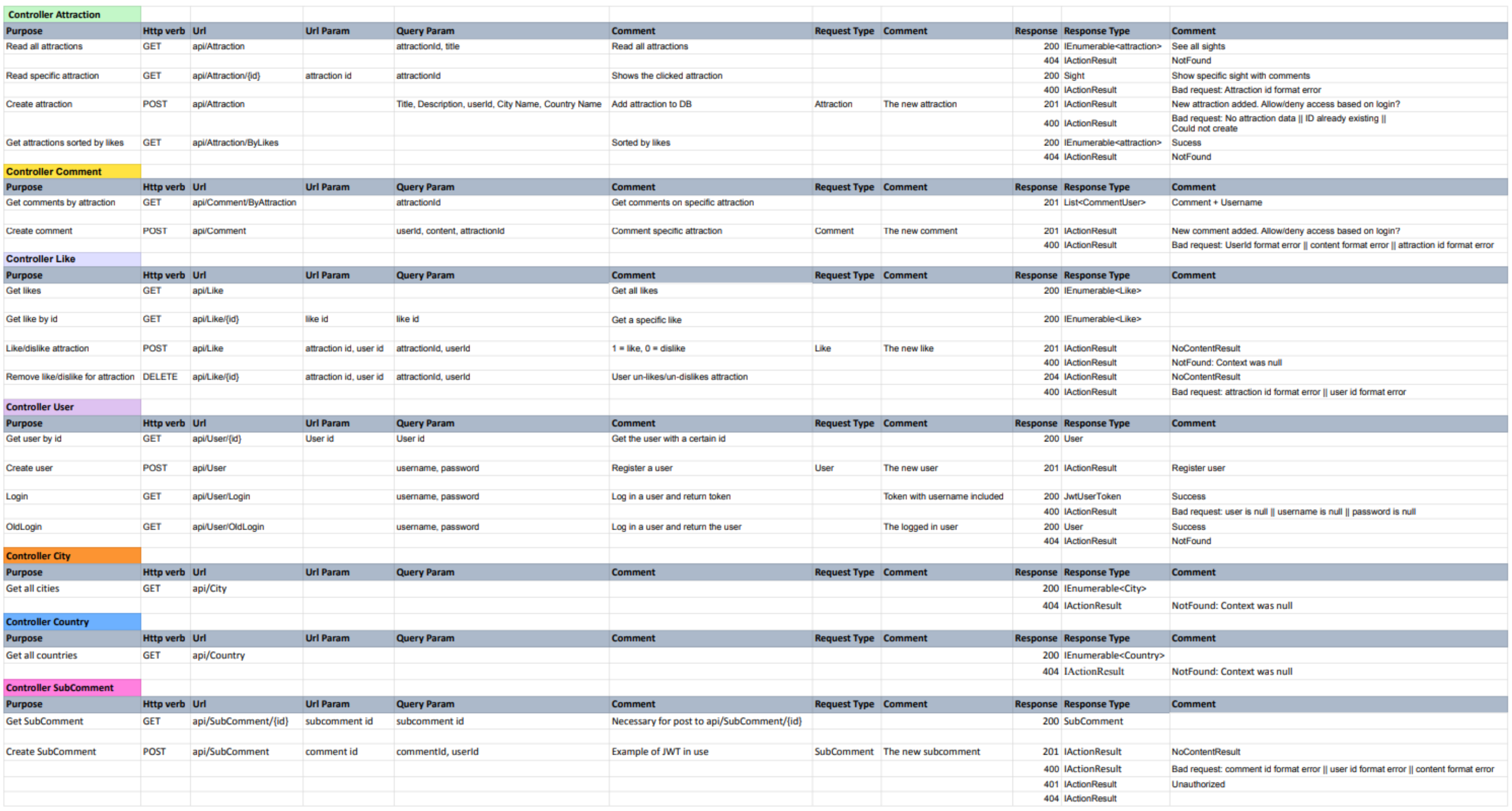
```
{
  "JsonWebTokenKeys": {
    "ValidateIssuerSigningKey": true,
    "IssuerSigningKey": "64A63153-11C1-4919-9133-EFAF99A9B456",
    "ValidateIssuer": true,
    "ValidIssuer": "https://localhost:7249",
    "ValidateAudience": true,
    "ValidAudience": "https://localhost:7249",
    "RequireExpirationTime": true,
    "ValidateLifetime": true
  }
}
```

Övergripande mjukvaruarkitektur

Filstruktur

Vi har lagt upp databasen, API:et och webbsidan i samma GitHub repo. API:et är uppdelat i olika controllers och models. Webbsidan är uppdelad i återanvändbara komponenter.

Web API:ets struktur



Controller Attraction

Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Read all attractions	GET	api/Attraction		attractionId, title	Read all attractions			200	IEnumerable<attraction>	See all sights
								404	IActionResult	NotFound
Read specific attraction	GET	api/Attraction/{id}	attraction id	attractionId	Shows the clicked attraction			200	Sight	Show specific sight with comments
								400	IActionResult	Bad request: Attraction id format error
Create attraction	POST	api/Attraction		Title, Description, userId, City Name, Country Name	Add attraction to DB	Attraction	The new attraction	201	IActionResult	New attraction added. Allow/deny access based on login?
								400	IActionResult	Bad request: No attraction data || ID already existing || Could not create
Get attractions sorted by likes	GET	api/Attraction/ByLikes			Sorted by likes			200	IEnumerable<attraction>	Sucess
								404	IActionResult	NotFound

Controller Comment

Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Get comments by attraction	GET	api/Comment/ByAttraction		attractionId	Get comments on specific attraction			201	List<CommentUser>	Comment + Username
Create comment	POST	api/Comment		userId, content, attractionId	Comment specific attraction	Comment	The new comment	201	IActionResult	New comment added. Allow/deny access based on login?
								400	IActionResult	Bad request: UserId format error || content format error || attraction id format error

Controller Like

Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Get likes	GET	api/Like			Get all likes			200	IEnumerable<Like>	
Get like by id	GET	api/Like/{id}	like id	like id	Get a specific like			200	IEnumerable<Like>	
Like/dislike attraction	POST	api/Like	attraction id, user id	attractionId, userId	1 = like, 0 = dislike	Like	The new like	201	IActionResult	NoContentResult
								400	IActionResult	NotFound: Context was null
Remove like/dislike for attraction	DELETE	api/Like/{id}	attraction id, user id	attractionId, userId	User un-likes/un-dislikes attraction			204	IActionResult	NoContentResult
								400	IActionResult	Bad request: attraction id format error || user id format error

Controller User

Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Get user by id	GET	api/User/{id}	User id	User id	Get the user with a certain id			200	User	
Create user	POST	api/User		username, password	Register a user	User	The new user	201	IActionResult	Register user
Login	GET	api/User/Login		username, password	Log in a user and return token		Token with username included	200	JwtUserToken	Success
								400	IActionResult	Bad request: user is null || username is null || password is null
OldLogin	GET	api/User/OldLogin		username, password	Log in a user and return the user		The logged in user	200	User	Success
								404	IActionResult	NotFound

Controller City

Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Get all cities	GET	api/City						200	IEnumerable<City>	
								404	IActionResult	NotFound: Context was null

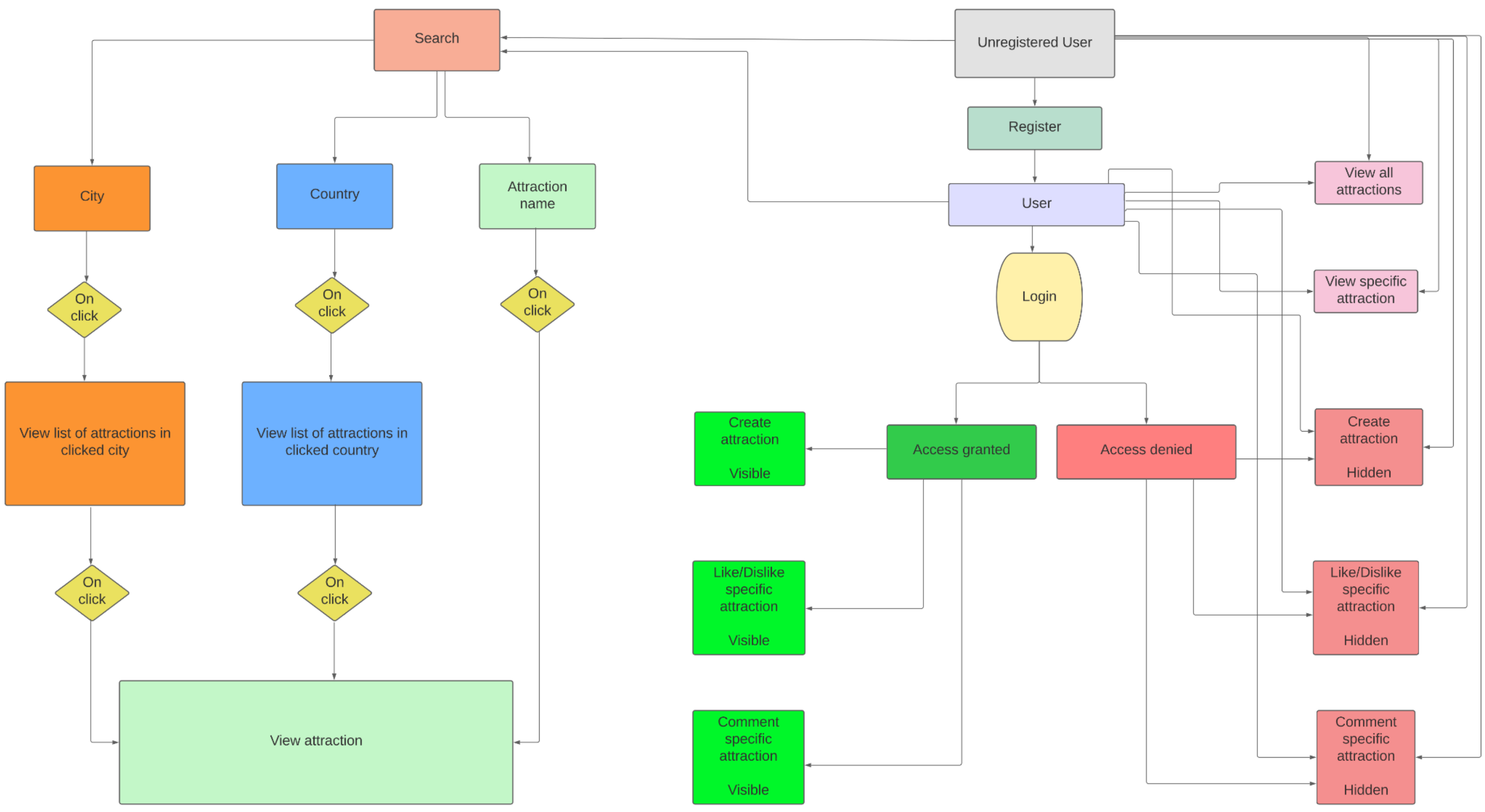
Controller Country

Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Get all countries	GET	api/Country						200	IEnumerable<Country>	
								404	IActionResult	NotFound: Context was null

Controller SubComment

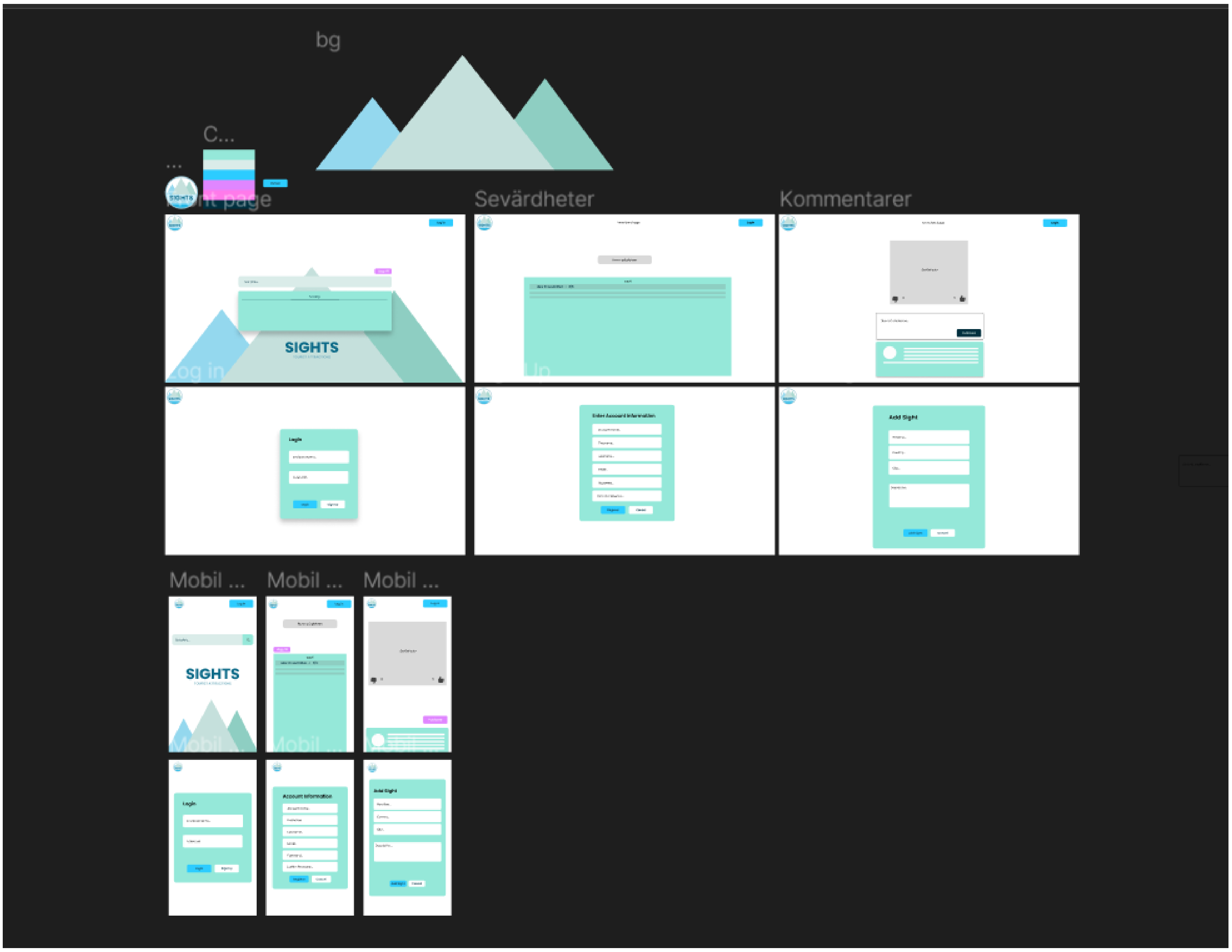
Purpose	Http verb	Url	Url Param	Query Param	Comment	Request Type	Comment	Response	Response Type	Comment
Get SubComment	GET	api/SubComment/{id}	subcomment id	subcomment id	Necessary for post to api/SubComment/{id}			200	SubComment	
Create SubComment	POST	api/SubComment	comment id	commentId, userId	Example of JWT in use	SubComment	The new subcomment	201	IActionResult	NoContentResult
								400	IActionResult	Bad request: comment id format error || user id format error || content format error
								401	IActionResult	Unauthorized
								404	IActionResult	

Sights flödesschema


Search
Unregistered User
Register
City
Country
Attraction name
User
View all attractions
On click
On click
On click
Login
View specific attraction
View list of attractions in clicked city
View list of attractions in clicked country
Create attraction Visible
Access granted
Access denied
Create attraction Hidden
On click
On click
Like/Dislike specific attraction Visible
Like/Dislike specific attraction Hidden
View attraction
Comment specific attraction Visible
Comment specific attraction Hidden

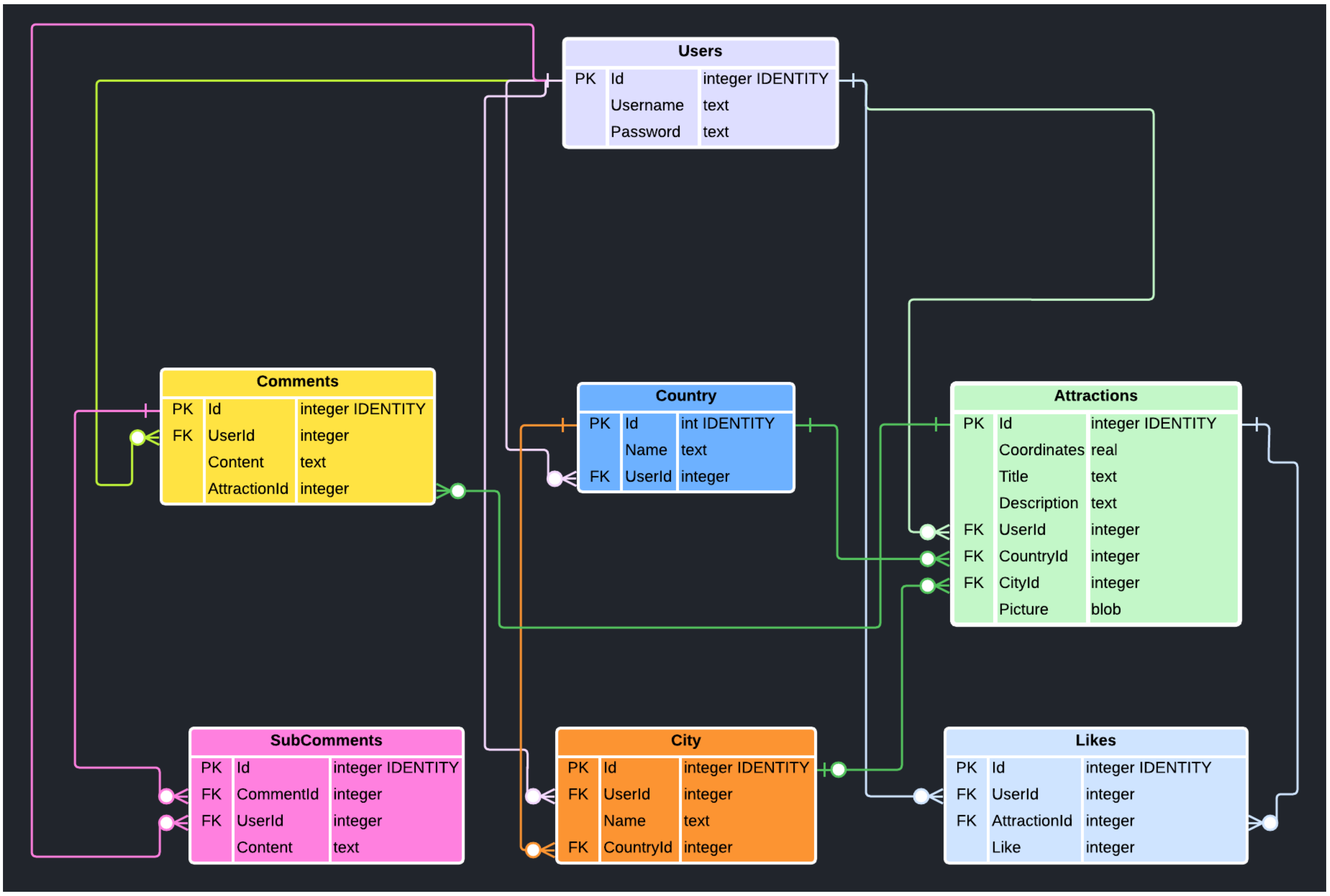
Figma för frontend

Figma användes för att på ett illustrativt sätt ta fram design samt struktur på hemsidan



Databasen arkitektur

Databasen är gjord i Sqlite. Detta betyder att det finns ett begränsat antal datatyper att välja på.



Users

PK	Id	integer IDENTITY
	Username	text
	Password	text

Comments

PK	Id	integer IDENTITY
FK	UserId	integer
	Content	text
	AttractionId	integer

Country

PK	Id	int IDENTITY
	Name	text
FK	UserId	integer

Attractions

PK	Id	integer IDENTITY
	Coordinates	real
	Title	text
	Description	text
FK	UserId	integer
FK	CountryId	integer
FK	CityId	integer
	Picture	blob

SubComments

PK	Id	integer IDENTITY
FK	CommentId	integer
FK	UserId	integer
	Content	text

City

PK	Id	integer IDENTITY
FK	UserId	integer
	Name	text
FK	CountryId	integer

Likes

PK	Id	integer IDENTITY
FK	UserId	integer
FK	AttractionId	integer
	Like	integer

Implementation av JWT

Vi har nu implementerat JWT överallt där vi använder POST-requests.