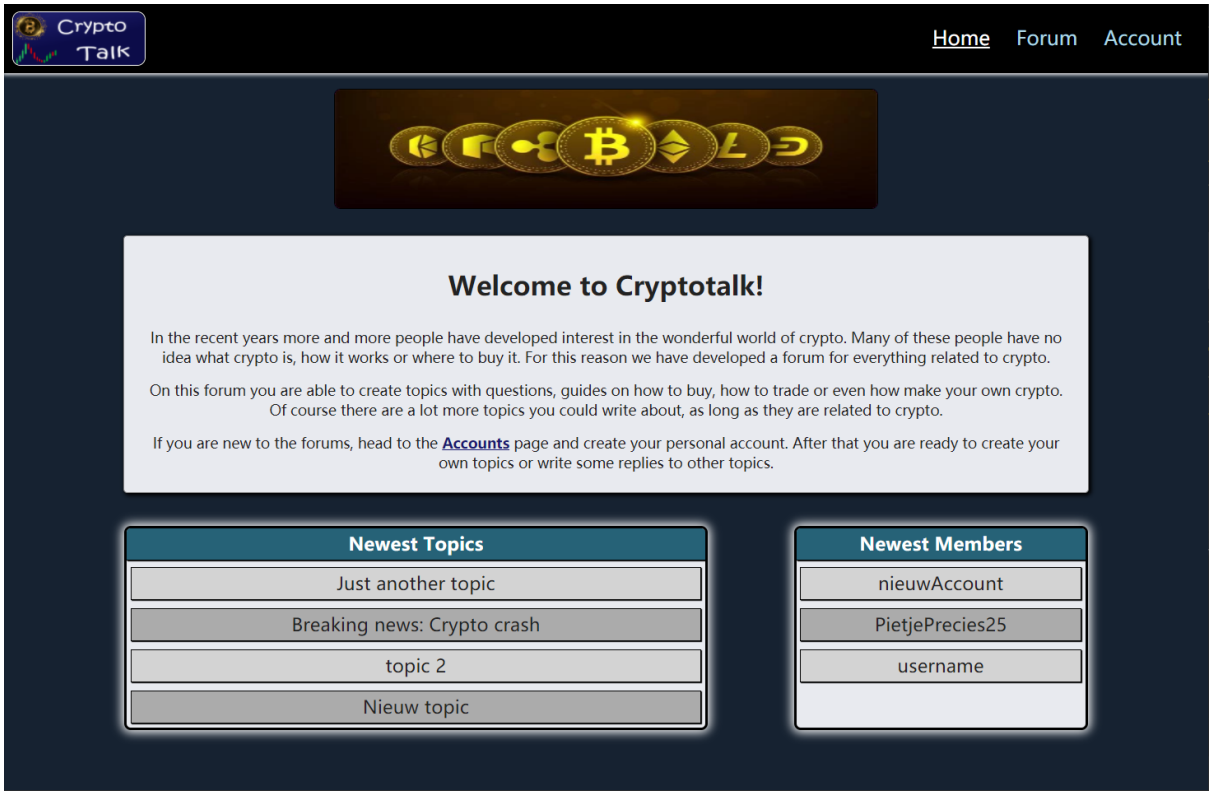
Documentatie webdesign

CryptoTalk



Klas: EHI1V.Sd
Edwin Heuver (139566)

Inhoudsopgave

Inleiding & ontwerpkeuzes

Dit verslag is geschreven naar aanleiding van de opdracht om een webpagina te maken in combinatie met de backend volgens het REST principe. In dit verslag is te een overzicht te vinden van de verschillende classes in de backend. Daarna zijn in het hoofdstuk “sequence diagrammen” twee diagrammen opgenomen om verduidelijking te geven in complexere aspecten van de backend. Opvolgend op de diagrammen is een overzicht te vinden van alle mogelijke API-requests. Ten slotte is in het hoofdstuk front-end een overzicht te vinden van wireframes voor de verschillende pagina’s.

Bij het ontwikkelen van dit project waren enkele ontwerpkeuzes die niet voor de hand liggen. Een voorbeeld hiervan is het ophalen van alle topics. Bij het ophalen van alle topics worden deze door de repository gelijk gesorteerd op aflopende volgorde van het moment dat ze gemaakt zijn. De reden hiervoor is dat ik graag de nieuwste topics bovenaan weer wil geven op de website. Dit had gekund door in javascript alle topics te ordenen, maar dit leek mij net zo makkelijk om met een query bij het ophalen uit de database te doen.
Een andere ontwerp keuze is dat op de forum pagina de aantallen van aangemaakte topics/members/posts gehardcode zijn. Hiervoor is gekozen omdat uiteindelijk onvoldoende tijd was om de bijbehorende backend calls te maken, de documentatie aan te passen en de front-end deze gegevens op te laten vragen.

Ook het API-design wekt mogelijk vragen op. De backend bevat volledige CRUD operaties voor de Post entiteit, echter wordt deze in de front end niet gebruikt. Het maken van een reply post wordt namelijk gedaan door deze toe te voegen aan een topic. Het ophalen van posts is ook niet nodig via de postcontroller, doordat posts al worden opgehaald wanneer een topic wordt opgevraagd. Ten slotte is er ook geen functionaliteit om posts te updaten/verwijderen in de front end. Ondanks dat posts via de font end gemaakt en opgevraagd kunnen worden, wordt hiervoor de PostController class niet gebruikt.

Ten slotte wil ik toelichting geven bij het interactiever maken van de pagina’s. Door een transition toe te voegen bij verschillende elementen zoals het hoveren over topics, de navigatiebalk of het logo in de navigatiebalk, voelt de website een stuk vloeiender aan. Ook al is de transition bij de meeste elementen slechts 500-800ms, doet het de interactie soepeler aanvoelen. Ook vond ik het aanmaken van een topic enigszins saai overkomen. Om deze reden heb ik met javascript geprogrammeerd dat wanneer op “create” gedrukt wordt, eerst de verschillende secties uit beeld glijden, voordat het scherm voor het aanmaken van een topic zichtbaar wordt.

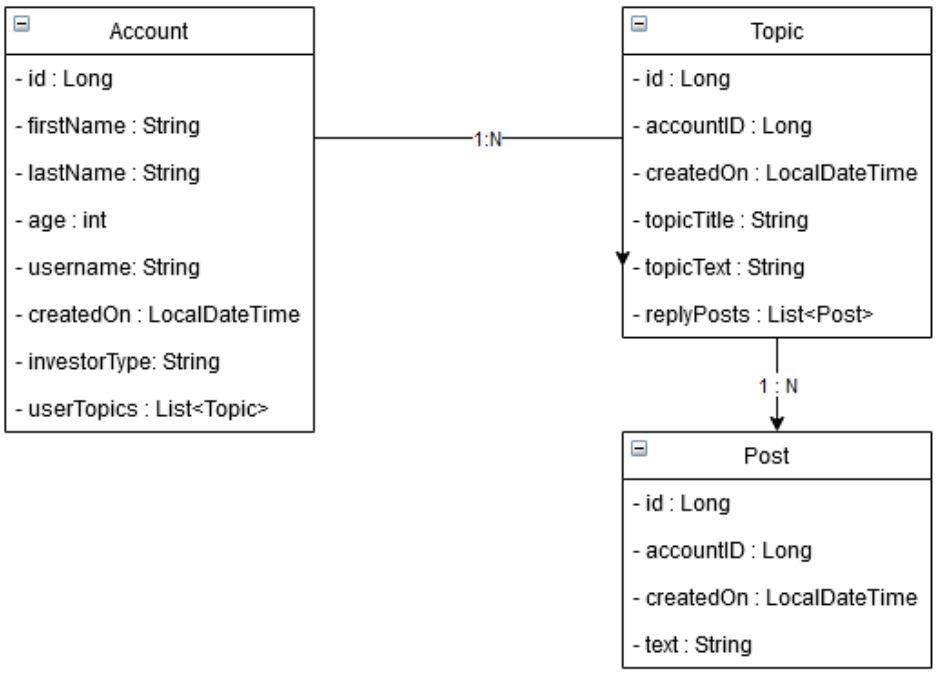
1. Class diagram

In Figuur 1 is het class diagram weergegeven van de backend van het forum. De backend bestaat uit drie classes genaamd “Account”, “Topic” en “Post”.

Een Account kan door een gebruiker worden aangemaakt met zijn naam, leeftijd, gebruikersnaam en type investeerder. Het id is een uniek veld dat automatisch wordt gegenereerd. Het veld “createdOn” wordt door de backend automatisch ingevuld met een LocalDateTime op het moment dat het account wordt aangemaakt. Ten slotte bevat elk account een lijst met Topics. Alle topics die door een gebruiker worden aangemaakt zullen in deze lijst opgeslagen worden.

De Topic class bevat net als een account het createdOn veld om te kunnen achterhalen wanneer deze gemaakt is. Daarnaast bevat het een uniek id en het id van het account dat de Topic heeft aangemaakt. Ook bevat een topic een titel in de vorm van een String en de Tekst die door de gebruiker is ingevuld bij het maken van de topic. Ten slotte bevat iedere Topic een lijst met Posts. Deze lijst kan leeg zijn, maar indien iemand een reply plaatst bij een topic zal de reply worden opgeslagen in deze lijst.

De Post class wordt gebruikt voor het opslaan van reacties op een Topic. Indien een reactie geplaats wordt krijgt deze een unieke id. Daarnaast wordt het id van de account welke de post heeft aangemaakt opgeslagen. Verder wordt van iedere post bijgehouden wanneer deze is aangemaakt en welke tekst het bevat.



Figuur 1 Class diagram

2. Sequence diagrammen

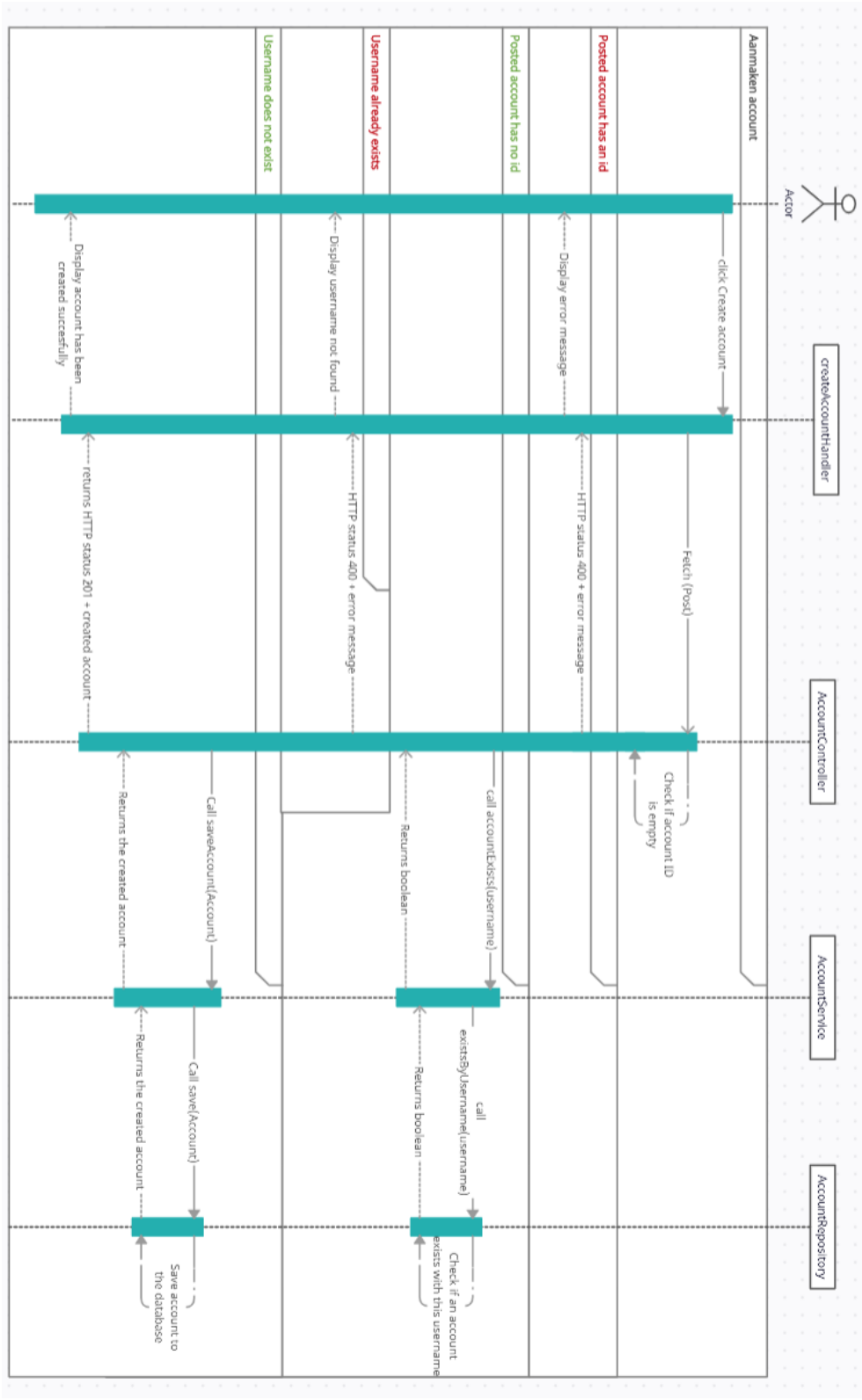
2.1 Maken van een account

In Figuur 2 is het sequence diagram weergegeven van het maken van een nieuw useraccount vanaf het punt dat de gebruiker alle velden op de website correct heeft ingevuld en op “create account” klikt. Wanneer hij dit doet wordt asynchroon een post gedaan met behulp van de fetch api. De backend vangt deze op in de AccountController class, waar wordt begonnen met het controleren van de body van de post.

De body van een account mag geen ID bevatten. Indien de body dit wel bevat wordt dit door de AccountController opgevangen, welke een HTTP status 400 terug stuurt met een bijbehorend bericht.

Indien geen ID is meegestuurd in de post, stuurt de AccountController een request naar de AccountService om te controleren of er een account bestaat met deze username. De service stuurt dit request door naar de AccountRepository, welke de database raadpleegt om dit te controleren. Vervolgens stuurt de repository een boolean terug naar de service, welke dit doorstuurt naar de AccountController. Indien de controller “true” ontvangt van de service, wordt een HTTP status 400 naar de createAccountHandler teruggestuurd met het bericht dat deze username reeds bestaat. De handler zal dit omzetten naar een bericht dat aan de gebruiker wordt weergegeven.

Indien de body correct is, zal de controller het verzoek doorsturen naar de AccountService class. Deze stuurt op zijn beurt een verzoek naar de AccountRepository om het account op te slaan in de database. Wanneer dit gelukt is wordt het account gereturned naar de Service en vervolgens naar de Controller. De controller zal vervolgens een HTTP status van 201 (created) naar de client sturen in combinatie met het zojuist aangemaakte account.

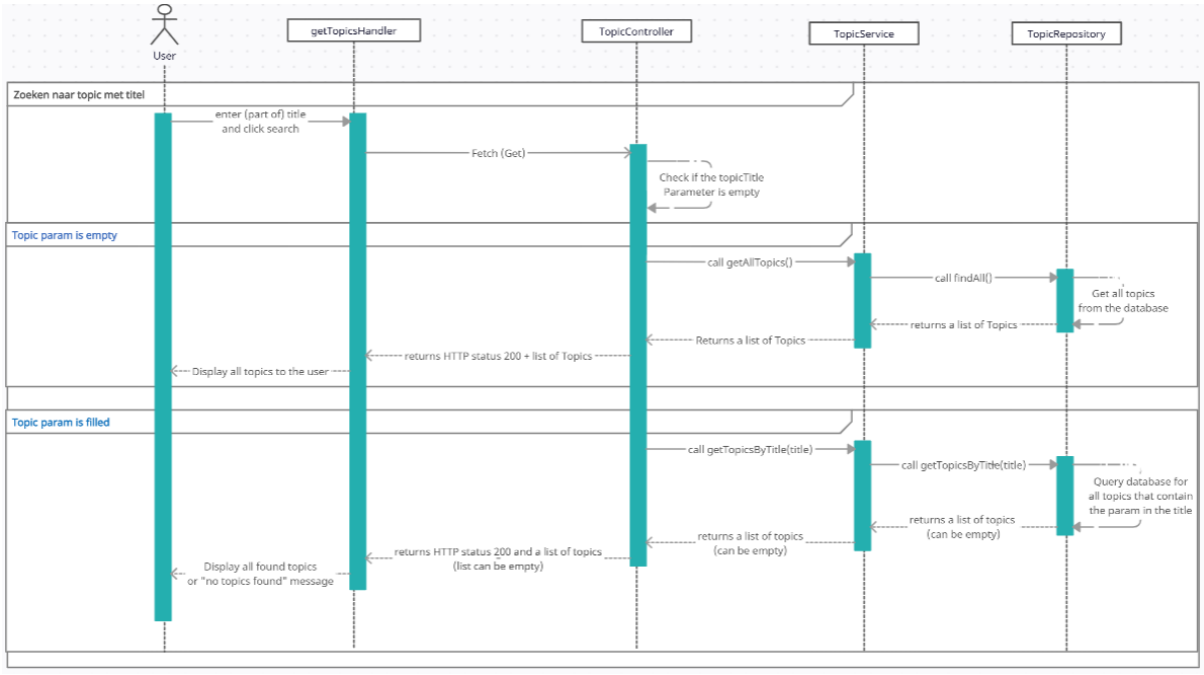


Figuur 2 Sequence diagram account creation

2.1 Opvragen van topics met specifieke titel

In Figuur 3 is de sequence diagram te zien voor het zoeken van een topic met een bepaalde titel. Op de website vult de gebruiker (een deel van) de titel in waar hij op wil zoeken. Met behulp van javascript wordt vervolgens een fetch op de topic controller gedaan. De topic controller controleert eerst of de titel parameter leeg is. Indien dit het geval is zal hij bij de topic service alle topics opvragen. De topic service linkt dit door naar de topicRepository, welke alle topics uit de database zal opvragen in aflopende volgorde van tijdstip van aanmaken. Vervolgens worden deze topics als lijst terug gestuurd naar de topicService, dan naar de topicController en vervolgens naar de website. Ten slotte worden alle topics aan de website toegevoegd. De reden dat deze in topics in aflopende volgorde gesorteerd worden, is zodat de nieuwste topics bovenaan weergegeven worden

Indien wel een zoek titel is opgegeven vraagt de topicController alle topics met een specifieke titel op bij de topicService. De service vraagt deze vervolgens op bij de topicRepository, welke met een query alle topics uit de database haalt die de zoekterm in de titel bevatten en deze aflopend sorteert op basis van moment van aanmaken. Vervolgens worden deze topics teruggestuurd en uiteindelijk weergegeven op de website. Indien geen topics gevonden zijn zal een bericht weergegeven worden dat er geen topics gevonden worden met de betreffende zoekterm.



Figuur 3 Sequence diagram Topic Get

3. API requests

3.1 GET requests

GET	**/accounts**		
Get a list of all user accounts			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	200	List of accounts, can be empty	
VOORBEELD	Get localhost:8080/accounts		

GET	**/accounts/{id}**		
Get an account by id			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Id *	Path	Filter on account id
Responses:	**Code**	**Description / example if successful**	
	200	The account with the given id	
	404	Thrown if no account exists with the given id	
VOORBEELD	Get localhost:8080/accounts/1		

GET	**/accounts/newest**		
Get a list of x amount of newest user accounts			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Amount*	Query	Amount of newest accounts to return
Responses:	**Code**	**Description / example if successful**	
	200	List of accounts, can be empty	
VOORBEELD	Get localhost:8080/accounts/newest?amount=5		

GET	/accounts/search		
Get an account by username			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	username *	Query	Username of the account to find
Responses:	**Code**	**Description / example if successful**	
	200	The account based on the username	
	404	Thrown if the account doesn't exist	
VOORBEELD	Get localhost:8080/accounts/search?username=cryptoguy		

GET	/posts/{id}		
Get a post by id			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Id *	Path	Filter on post id
Responses:	**Code**	**Description / example if successful**	
	200	The post with the given id	
	404	Thrown if no post exists with the given id	
VOORBEELD	Get localhost:8080/posts/1		

GET	/topics		
Get a list of all topics in descending order of creation date/time			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	topicTitle	Query	Filter on topicTitle (case sensitive)
Responses:	**Code**	**Description / example if successful**	
	200	A list of topics, can be empty	
VOORBEELD	Get localhost:8080/topics -> returns all topics in descending order Get localhost:8080/topics?topicTitle=cat -> returns all topics with cat in the title in descending order of creation;		

GET	/topics/newest		
Get a list of x amount of newest topics			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Amount *	Query	Amount of newest topics to return
Responses:	**Code**	**Description / example if successful**	
	200	A list of topics, can be empty	
VOORBEELD	Get localhost:8080/topics/newest?amount=5		

GET	/topics/{id}/posts		
Get a list of all reply posts on a specific topic			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Id*	path	Id of the topic
Responses:	**Code**	**Description / example if successful**	
	200	List of all reply posts on this topic, can be empty	
	404	Thrown if no topic with this id can be found	
VOORBEELD	Get localhost:8080/topics/1/posts		

3.2 POST requests

POST	/accounts		
Create a new account			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Account*	body	The Account to add
Responses:	**Code**	**Description / example if successful**	
	201	Returned when project was made successfully	
	400	Thrown if something went wrong with the account body or if the username already exists	

VOORBEELD	POST localhost:8080/accounts
	Body:
	{
	"firstName": "voornaam",
	"lastName": "achternaam",
	"username": "cryptoguy",
	"age": 20,
	"investorType": "Whale"
	}		

POST	/topics		
Create a new topic			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Topic*	body	The topic to add
Responses:	**Code**	**Description / example if successful**	
	201	Returned when the Topic was made successfully	
	400	Thrown if something went wrong with the topic body	

VOORBEELD	POST localhost:8080/topics
	Body:
	{
	"accountID": 1,
	"topicTitle": "My first topic",
	"topicText": "My first topic about crypto"
	}		

POST	/topics/{id}/posts		
Create a new post and assign it to a topic			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id *	path	Id of the topic to add the post to
	post*	body	The post to add to the topic
Responses:	**Code**	**Description / example if successful**	
	201	Returned if the post was successfully added to the topic	
	404	Thrown if there is no topic with the given id	
	400	Thrown if something went wrong with the post body	

VOORBEELD	POST localhost:8080/topics/1/posts
	Body:
	{
	"accountID": 1,
	"text": "Hello, this is my first reply"
	}		

3.3 PUT requests

PUT	**/accounts/{id}**		
Update an account by ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	ID of the account to update
	Account *	body	Updated Account
Responses:	**Code**	**Description / example if successful**	
	200	Returned if the update was successful	
	400	Thrown if the body was wrong or if the id in the path does not match the id in the updated account	
	404	Thrown if the Account with this id cannot be found	

VOORBEELD	PUT localhost:8080/accounts/1
	Body:
	{
	"firstName": "nieuwe voornaam",
	"lastName": "nieuwe achternaam",
	"username": "cryptoguy",
	"age": 20,
	"investorType": "Whale"
	}		

PUT	**/topics/{id}**		
Update a topic with a specific id			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Id *	Path	The id of the topic to update
	Topic *	Body	The updated topic
Responses:	**Code**	**Description / example if successful**	
	200	Returned if the update was successful	
	400	Thrown if the body was wrong or if the id in the path does not match the id in the body	
	404	Thrown if there is no topic with this id	

VOORBEELD	PUT localhost:8080/topics/1
	Body:{
	"accountID": 1,
	"topicTitle": "My first topic",
	"topicText": "Updated topic text"
	}		

<table>
<tr><td>PUT</td><td colspan="3">/posts/{id}</td></tr>
<tr><td colspan="4">Update a post with a specific id</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Parameters:</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td rowspan="2">Add a * to the name of required parameters.</td><td>Id *</td><td>Path</td><td>The id of the post to update</td></tr>
<tr><td>Post *</td><td>Body</td><td>The updated post</td></tr>
<tr><td>Responses:</td><td>Code</td><td colspan="2">Description / example if successful</td></tr>
<tr><td rowspan="3"></td><td>200</td><td colspan="2">Returned if the update was successful</td></tr>
<tr><td>400</td><td colspan="2">Thrown if the body was wrong or if the id in the path does not match the id in the body</td></tr>
<tr><td>404</td><td colspan="2">Thrown if there is no post with this id</td></tr>
<tr><td>VOORBEELD</td><td colspan="3">PUT localhost:8080/posts/1
Body:
{
"accountID": 1,
"text": "Hello, this is my first reply but now updated"
}</td></tr>
</table>

3.4 DELETE requests

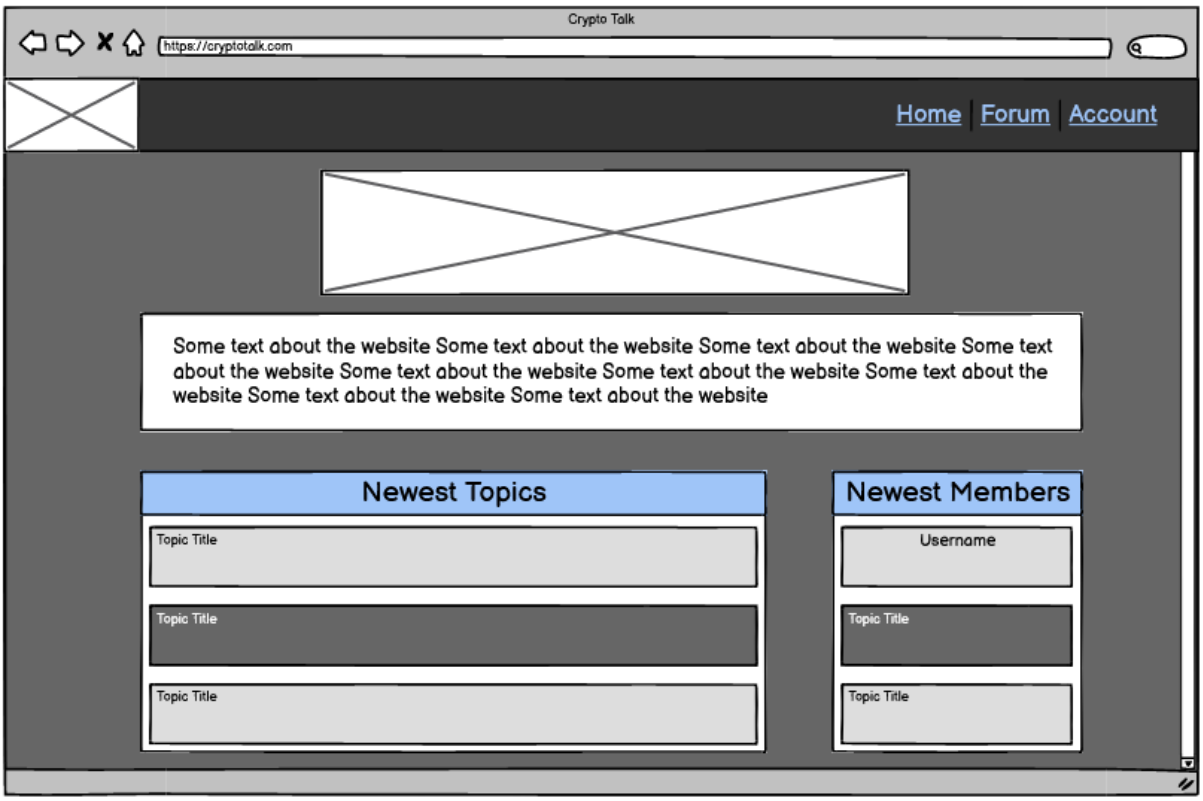
DELETE	**/accounts/{id}**		
Delete an account with a specific ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id *	path	The id of the account to delete
Responses:	**Code**	**Description / example if successful**	
	204	Returned if the account was successfully deleted	
	404	Thrown if the account does not exist	
VOORBEELD	DELETE localhost:8080/accounts/1		

DELETE	**/topics/{id}**		
Delete a topic with a specific ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id *	path	The id of the topic to delete
Responses:	**Code**	**Description / example if successful**	
	204	Returned if the topic was successfully deleted	
	404	Thrown if the topic does not exist	
VOORBEELD	DELETE localhost:8080/topics/1		

DELETE	**/posts/{id}**		
Delete a post with a specific ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id *	path	The id of the post to delete
Responses:	**Code**	**Description / example if successful**	
	204	Returned if the post was successfully deleted	
	404	Thrown if the post does not exist	
VOORBEELD	DELETE localhost:8080/posts/1		

4. Front-end

4.1 Homepage



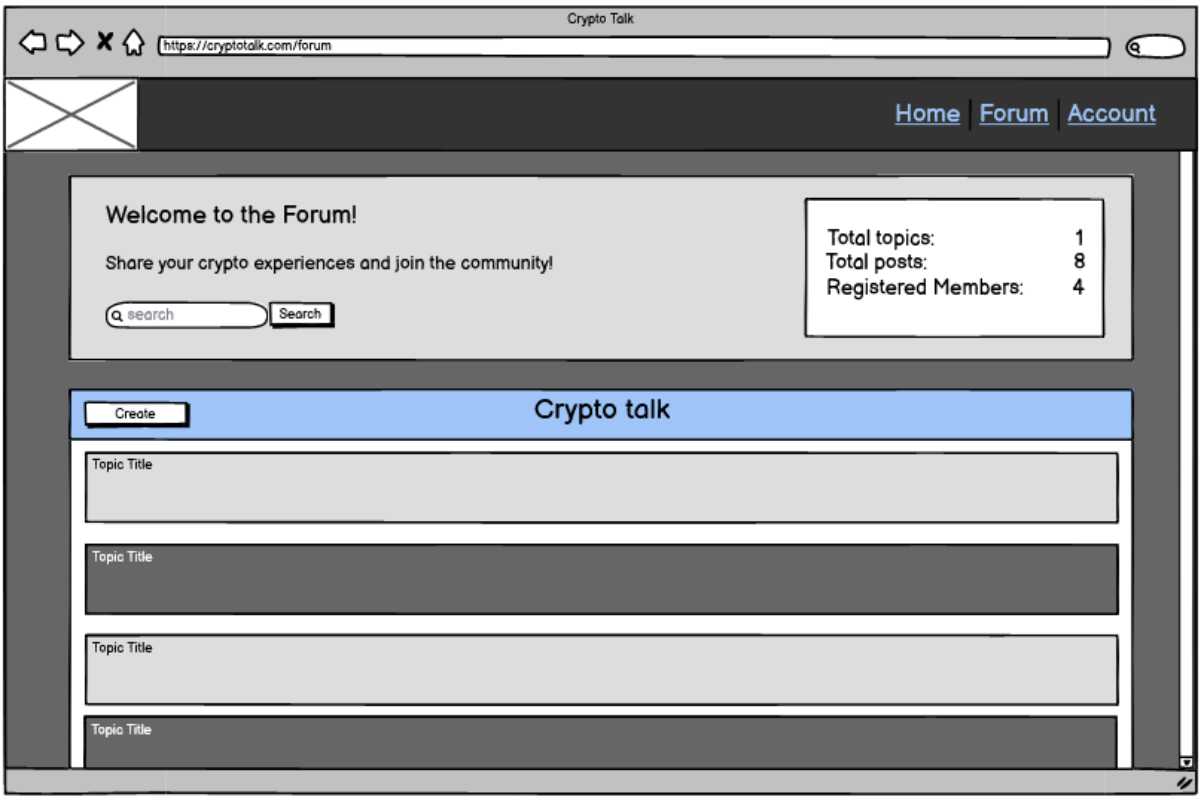
Figuur 4 wireframe homepage

In figuur 4 is het wireframe te zien van de homepage van het forum. Links boven in de header is ruimte voor een logo, welke indien er op geklikt wordt naar de homepage leidt. Verder is in de header een navigatiebalk weergegeven welke naar de verschillende andere pagina's leidt. Deze header is voor iedere wireframe hetzelfde en zal dus bij de overige wireframes niet opnieuw behandeld worden.

Op de pagina zelf is bovenaan een rechthoek met plek voor een banner. Onder de banner is ruimte voor tekst met uitleg over de website. Ten slotte zijn onderaan de pagina secties weergegeven waarin de (max 5) nieuwste topics en members worden weergegeven. Er kan op de topics geklikt worden om het betreffende topic te bekijken. De nieuwe members kunnen niet bekeken worden door te klikken.

Hoewel de wireframe voornamelijk in grijstinten is weergegeven is de daadwerkelijke website voornamelijk in donkerblauw/lichtblauwe tinten gemaakt.

4.2 Forum



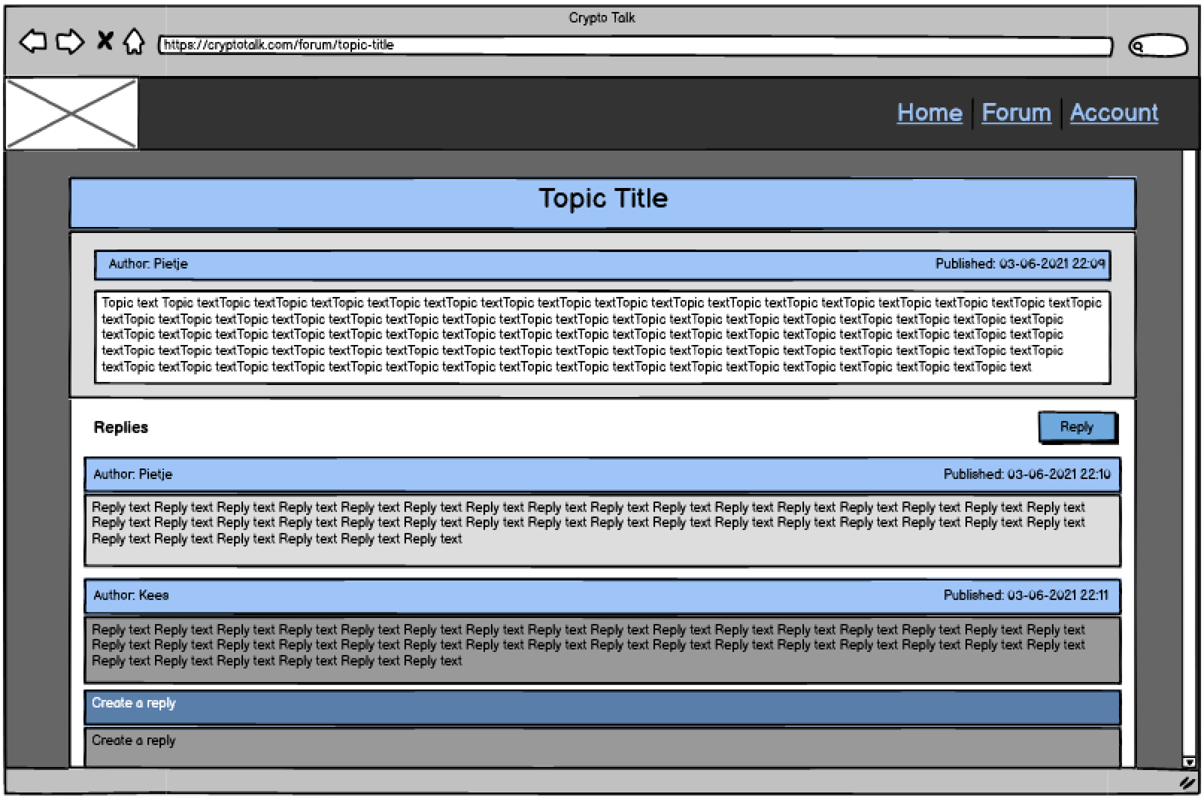
Figuur 5 wireframe forum

Op de forum pagina is een welkom sectie waarin de gebruiker welkom geheten wordt op het forum. Onder het welkomstbericht is een zoekfunctie waarmee de topics gefilterd kunnen worden op (een deel van) de titel. Deze zoekfunctie is hoofdletter gevoelig. Rechts van het welkom bericht is een overzicht van het totaal aantal gemaakte topics, reply posts en het aantal geregistreerde accounts. Helaas was er geen tijd meer om dit te implementeren waardoor de daadwerkelijke aantallen zijn gehardcode.

Onder het welkombericht is een overzicht te zien van alle topics die zijn aangemaakt. Deze topics zijn gesorteerd op aflopende volgorde van het moment van plaatsen, waardoor de nieuwe topics bovenaan weergegeven zullen worden. Indien je met de muis over een topic beweegt zal deze donker kleuren om aan te geven dat er op geklikt kan worden.

Ook is er een “create” button in de linker bovenhoek van de topic container weergegeven. Indien hier op geklikt wordt zal een animatie plaatsvinden waardoor het welkom gedeelte naar links uit beeld schuift, vervolgens het topics gedeelte naar boven uit beeld schuift, waarna een scherm zichtbaar wordt waarin een nieuwe topic aangemaakt kan worden.

4.3 Forum Topic



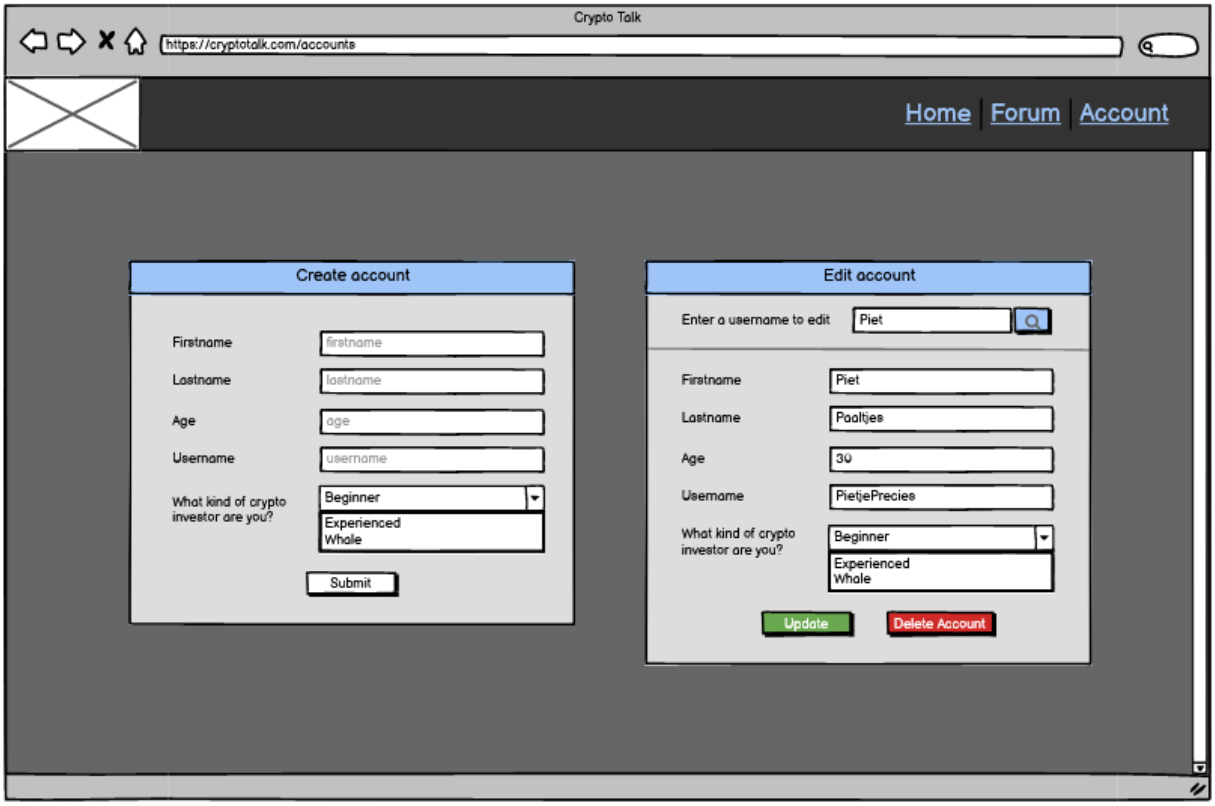
Figuur 6 wireframe topic

Wanneer op de homepage of in het forum op een topic geklikt wordt, navigeert de gebruiker naar de betreffende topic. Op deze pagina is bovenaan de topic weergegeven met de auteur, datum van plaatsen en de geplaatste tekst.

Onder de topic is een lijst van alle reacties op deze topic weergegeven met de auteur van de reactie en wanneer deze geplaatst is. Indien er geen reacties zijn wordt weergegeven dat nog niemand gereageerd heeft.

Rechts onder de topic staat de “reply” knop. Indien op deze knop gedrukt wordt zal de topic niet langer zichtbaar zijn en wordt een scherm weergegeven waar de reactie tekst en de gebruikers username ingegeven kan worden. Indien hij op “create” drukt wordt de reply geplaatst, indien hij op “cancel” drukt wordt niets gepost. In beide gevallen zal de topic weer zichtbaar worden.

4.4 Accounts



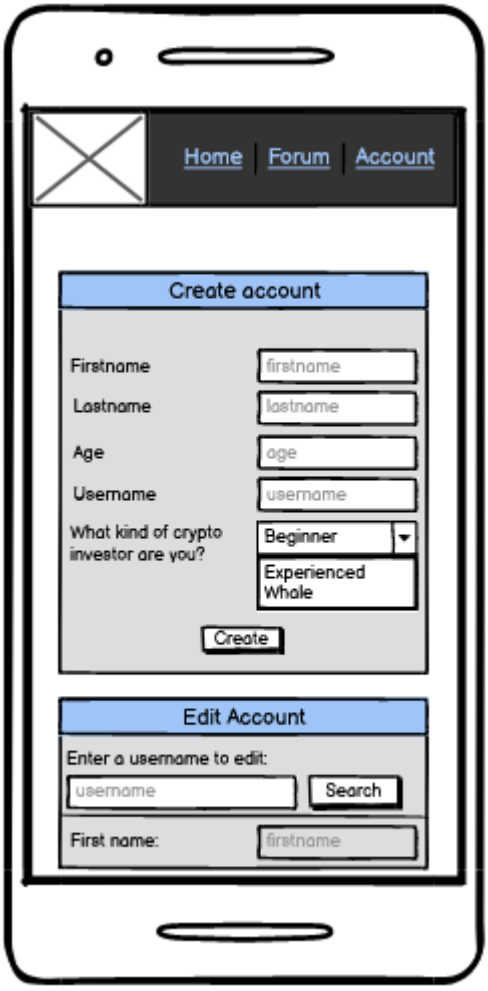
Figuur 7 wireframe accountspagina

In figuur 7 is het wireframe te zien van de Account pagina. Aan de linkerzijde is de sectie te zien voor het aanmaken van een account en aan de rechterkant voor het aanpassen van een account.

Het aanmaken van een account spreekt voor zich. Indien op submit gedrukt wordt, zal met javascript gecontroleerd worden of de velden daadwerkelijk zijn ingevuld. Als dit niet het geval is zal er een foutmelding weergegeven worden. Als de velden wel correct zijn ingevuld zal geprobeerd worden om een fetch post te sturen naar de backend. Wanneer nog geen account met deze username bestaat, krijgt de gebruiker een melding dat het account is aangemaakt. Als het niet goed gaat komt er een melding waarin wordt weergegeven wat er fout is gegaan.

Bij het aanpassen van een account dient de gebruiker eerst het account op te zoeken dat hij/zij wil aanpassen. Dit gebeurt op basis van de gebruikersnaam. Indien de gebruikersnaam bestaat, zullen de velden ingevuld worden met de huidige gegevens van dit account. Vervolgens kan de gebruiker de velden aanpassen en de wijzigingen opslaan naar de database door op “edit” te drukken. De andere optie die de gebruiker heeft is om op delete te drukken. Indien hij/zij dit doet, zal de gezochte gebruiker worden verwijderd uit de database. In beide gevallen krijgt de gebruiker een melding over het resultaat.

4.5 Accounts – mobile



Figuur 8 wireframe accounts - mobiele versie

In figuur 8 is de wireframe voor de accounts pagina te zien wanneer deze wordt weergegeven op kleine schermen. De functionaliteit is hetzelfde als in het vorige hoofdstuk is behandeld, alleen zal de tekst en image in de header kleiner zijn. Ook zullen nu de create en edit sectie onder elkaar worden weergegeven in plaats van naast elkaar.