IPR Merijn Couweleers

Technische Informatica 2.1

Contents

[Analysefase: 2](#_Toc86611812)

[Beschrijving: 2](#_Toc86611813)

[MoSCoW: 2](#_Toc86611814)

[Must Have: 2](#_Toc86611815)

[Should Have: 2](#_Toc86611816)

[Could Have: 2](#_Toc86611817)

[Would Have: 2](#_Toc86611818)

[Technieken en Tooling: 2](#_Toc86611819)

[IDE 2](#_Toc86611820)

[Version Control 2](#_Toc86611821)

[JSON 3](#_Toc86611822)

[Ontwerpfase 3](#_Toc86611823)

[Realisatiefase 3](#_Toc86611824)

[Testen 3](#_Toc86611825)

[Bijlages: 4](#_Toc86611826)

[Netwerkprotocoldocument: 4](#_Toc86611827)

[KlassenDiagram 7](#_Toc86611828)

# Analysefase:

## Beschrijving:

Een server houdt projecten met pages bij. Deze projecten kunnen door clients worden opgevraagd en bekeken. Als de client de eigenaar is van dit project, kan deze het project aanpassen, door middel van pages toe te voegen, weg te halen of aan te passen. Wanneer de page na de aanpassingen wordt geüpload, past de server de veranderingen toe op de correcte versie.  
De server houdt de historie van projecten bij, de client kan dan een versie kiezen. Alleen de nieuwste versie kan worden aangepast door de eigenaar van het project.  
Een client kan een account aanmaken, waarmee deze dan projecten kan aanmaken, en uploaden op de server. Meerdere accounts kunnen aan een project werken, als collaborators. Een account bestaat uit een username en password. Usernames kunnen maar één keer voorkomen.  
Wanneer de client op de server hoofdpagina zit, kan deze alle projecten zien, en sorteren op verschillende gegevens. Het openen van een project laat de hoofdpage zien, met mogelijke verwijzingen naar andere pages.  
Een page kan verschillende elementen bevatten, zoals image, button, text. Deze kunnen door de eigenaar worden gesleept naar de gewenste locatie.

## MoSCoW:

### Must Have:

* TCP-server
* Inlog systeem
* File editing

### Should Have:

* Version control
* Multi user editing

### Could Have:

* Line Locking
* Caching van opgevraagde bestanden

### Would Have:

* Download/upload feature

## Technieken en Tooling:

## IDE

Als IDE is Visual Studio gebruikt, dit omdat het de gebruiker in staat stelt om de code te compileren, te runnen, en te debuggen. Hiernaast levert Visual Studio ook nog andere features, zoals het live sharen van de code. Hiermee kan er snel in tweetallen worden gedebugged.

## Version Control

Voor version control is gekozen voor Git. De service is geleverd door github, en als tool voor makkelijke communicatie met github is gekozen voor gitkraken.

## JSON

Er is gekozen om vooral met JSON te werken, hiervoor is de .NET library Newtonsoft gebruikt, waarmee o.a. het serialiseren wordt afgehandeld. Hierbij moet fileIO/netwerk transfer nog zelf worden gemaakt.

# Ontwerpfase

Het netwerkprotocoldocument en de klassediagram zijn beide als bijlage te vinden aan het einde van dit document. Hier zal ook een beschrijving bij staan.

# Realisatiefase

Tijdens het programmeren is regelmatig terug gekeken naar eerder geschreven code, en zo nodig geoptimaliseerd. Hierdoor is de code op globaal niveau qua kwaliteit gelijk gebleven. De code is veel getest, en gemaakt met een oog op features kunnen toevoegen. De code is goed leesbaar, en zo nodig opgedeeld in methode om de leesbaarheid te behouden. Voor variabelenamen zijn logische namen gekozen, en bij onduidelijke punten zijn comments toegevoegd met extra uitleg.

De code zit vrij dicht bij het ontwerp, echter kwamen er enkele complicaties waardoor er licht van het ontwerp moest worden afgeweken. De ontwerpen zijn toen aangepast om deze veranderingen te weergeven.

# Testen

De gebruikte manier om te testen is met behulp van unit tests, en zelf tijdens het programmeren te kijken of alles werkt zoals gewenst. Hierdoor is het mogelijk om snel tests uit te voeren, en in orde maken als het niet goed is.

Deze tests zijn goed uitgevoerd, en hebben geholpen met een goed product te leveren.

# Bijlages:

## Netwerkprotocoldocument:

Bericht format

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bytes | 2 | 1 | n | 2 | 1 |
| Waarde | Message length | Type | Message | Segment ID, 0 if final | Checksum of all other fields |

Type IDs:

Login: 0  
Request make account: 1  
Make account: 2  
  
Create project: 11  
Create page:12  
  
Request pages:20  
Request page: 21  
Upload page: 22  
Request change page: 23  
Upload changed page: 24  
  
Response ok: 40  
Response not ok: 41  
Request pages response: 42  
Request page response: 43  
  
Ping from client: 193  
Ping from server: 194  
  
Disconnect: 255

messages:

Login:

{

“username”: “<username>”,

“password”: “<encrypted password>”

}

Request make account:

request make account

Make account:

{

“username”: “<username>”,

“password”: “<encrypted password>”

}

Create project:

{

“projectName”: “<projectName>”

}

Create page:

{

“pageName”: “<pageName>”

}

Request pages:

request pages

Request page:

{

“page”: “<page ID>”

}

Upload page:

{

“elements”: […]

}

Request change page:

request change page

Upload changed page:

{

“page”: “<page ID>”

“elements”: […]

}

Response ok:

affirmative

Response not ok:

negative

Request pages response:

{

“Elements”: […]

}

Ping from client:

{

“sendtime”: <unix time>

}

Ping from server:

{

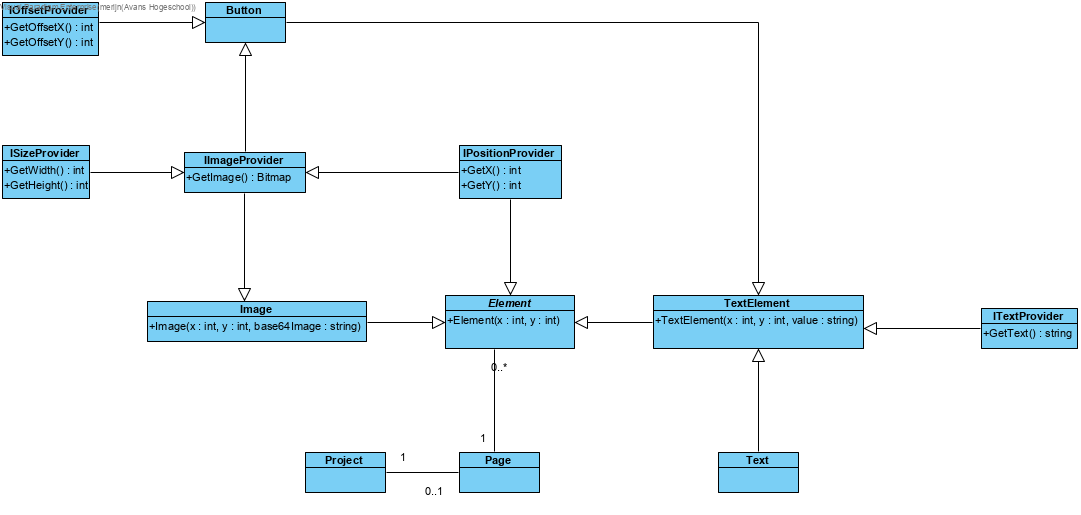
“sendtime”: < unix time>

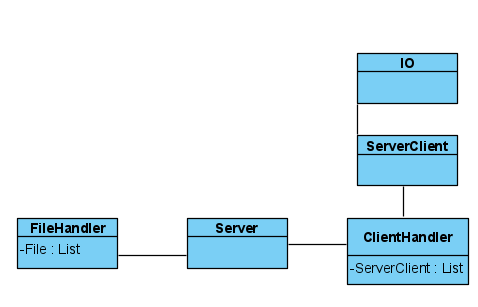
}

Disconnect:

disconnect

## KlassenDiagram

  
Dit is de layout voor een project, een project heeft 1 of meer pages. Deze pages hebben 1 of meer elementen. Deze elementen zijn van het type element, maar kunnen vanaf daar worden ge-extend om verdere functionaliteit toe te voegen.



Dit is de layout voor de server, hier staan de clientHandlers met serverClients, deze hebben IO voor hun bestanden, ook is er een algemene fileHandler voor niet client gerelateerde fileIO.