



Odisee Gent

Gebroeders de Smetstraat 1, 9000 Gent

MineColonies Automation

MineColonies Automation

Pjotr Brunain, Thibe Provost

Luca Vandenweghe, Jonas Van Kerkhove

MineColonies Automation

Inhoud

Inhoud	4
Codefragmentenlijst.....	7
Figurenlijst.....	8
Afkortingenlijst.....	9
Inleiding.....	10
1 Situering	11
1.1 Wat is MineColonies?	11
1.2 Wat is het probleem?	11
1.3 Wat is de oplossing?	11
2 Minecraft en MODS	12
2.1 Overzicht van de gebruikte Minecraft MODS	12
2.1.1 MineColonies.....	12
2.1.2 Applied Energistics 2.....	12
2.1.3 CC: Tweaked	12
2.1.4 Advanced Peripherals	12
2.2 LUA en Minecraft	13
2.2.1 LUA	13
2.2.2 LUA in CC: Tweaked	13
2.3 Extraheren van takenlijst uit MineColonies.....	14
2.3.1 Extraheren algemene taken	14
2.3.2 Extraheren buildertaken	14
2.4 Commando's.....	15
2.4.1 Inlezen en uitvoeren van gegeven commando's.....	16
3 C#-programma	17
3.1 Extractie van colonydata en storedata uit Minecraftfolder.....	17
3.1.1 Colonydata	17
3.1.2 storage	18
3.2 C# authenticatie	19
3.3 Extract recipes uit Minecraft.....	20
3.4 Commando's.....	21
4 Websiteontwikkeling	23
4.1 Overzicht van het Ontwikkelingsproces	23
4.2 Ontwerp met Figma	23
4.2.1 Flowchart en doelen.....	23

4.2.2	Eerste Figma schetsen & kleurenpalet selectie	24
4.2.3	Feedback en revisie	25
4.3	Technologiekeuze: Javascript, HTML en SCSS	26
4.3.1	Voordelen van Javascript	26
4.3.2	Structuur met HTML	27
4.3.3	Styling met SCSS.....	27
4.4	Overstap naar Nuxt framework	28
4.4.1	Wat is Nuxt?	28
4.4.2	Installatie en configuratie van Nuxt	28
4.4.3	Uitdagingen en oplossingen.....	28
4.5	Gebruik van Three.js	29
4.5.1	Wat is Three.js?.....	29
4.5.2	Leren en implementeren van Three.js	29
4.5.3	Interactieve Minecraft Steve	29
4.5.4	Integratie met Nuxt.....	29
4.6	Data-integratie en dashboard ontwikkeling	29
4.6.1	Koppeling met de API	29
4.6.2	Weergave van data op het dashboard	30
4.6.3	Data-ophaalmethode en voorbeeldcode.....	30
4.7	Backendontwikkeling	31
4.7.1	Beveiliging	31
4.7.2	Laravel problemen	31
5	Werking API.....	33
5.1	POST requests	33
5.1.1	Werelden	33
5.1.2	Colonies	33
5.1.3	Requests	34
5.1.4	Builderrequests	35
5.1.5	Storage items.....	36
5.1.6	Recipes items	37
5.2	GET requests	38
5.2.1	Werelden.....	38
5.2.2	Colonies	39
5.2.3	Recipes	40
5.3	PUT requests	41
5.4	DELETE requests.....	41

Conclusie	42
Handleiding	43
Literatuurlijst	49
Bijlagenoverzicht.....	50
Bijlage 1: Logboek rapporteren	50
Bijlage 2: Verslag1	51
Bijlage 3: Verslag2.....	52
Bijlage 4: Verslag3.....	53
Bijlage 5: Verslag4.....	54
Bijlage 6: Verslag5.....	55
Bijlage 6: Verslag6.....	56

Codefragmentenlijst

Codefragment 1: Initialize van WrapPeripherals	13
Codefragment 2: Extraheren algemene taken	14
Codefragment 3: extraheren van buildertaken	15
Codefragment 4: voorbeeld van een commando	16
Codefragment 5: extraheren van colonydata	17
Codefragment 6: extraheren van storedata	18
Codefragment 7: ververs refreshtoken	19
Codefragment 8: GetModsPaths	20
Codefragment 9: Extract recipe recursie	21
Codefragment 10: HTML-structuur voor zoekmachine optimalisatie	27
Codefragment 11: _vars.sass file met variabelen	27
Codefragment 12: Javascript implementatie	29
Codefragment 13: Javascript import statement	29
Codefragment 14: data weergave van de opgehaalde werelden	30
Codefragment 15: Asynchrone fetch van de werelden	30
Codefragment 16: POST Request body wereld	33
Codefragment 17: POST Request body colony	33
Codefragment 18: POST Request body request	34
Codefragment 19: POST Request body builder requests	35
Codefragment 20: POST Request body Storage items	36
Codefragment 21: POST Request body recipe items	37
Codefragment 22: Response body werelden	38
Codefragment 23: Response body colonies	39
Codefragment 24: Response body Recipes	40
Codefragment 25: PUT requests	41

Figurenlijst

Figuur 1: Gemeenschap in MineColonies	11
Figuur 2: Lijst van resources.....	11
Figuur 3: data flowchart	11
Figuur 4: ME Controller [2]	12
Figuur 5: Advanced Computer [3]	12
Figuur 6: Colony Integrator [13]	12
Figuur 7: ME Bridge [13]	12
Figuur 8: Figma flowchart website.....	24
Figuur 9: Kleurenpalet	25
Figuur 10: Figma design na feedback.....	25
Figuur 11: Burger menu.....	26
Figuur 12: Databank ontwerp	31
Figuur 13: Warehouse [11]	43
Figuur 14: Courier [11]	43
Figuur 15: Warehouse met storage bus	43
Figuur 16: Advanced monitor [3].....	44
Figuur 17: colony integrator [13].....	44
Figuur 18: ME interface [14]	44
Figuur 19: ME-Bridge [13].....	44
Figuur 20: Advanced computer [3].....	44
Figuur 21: Setup vooraanzicht.....	44
Figuur 22: Setup bovenaanzicht	44
Figuur 23: Website home page.....	45
Figuur 24: Unzipped folder	45
Figuur 25: Instance selectie	46
Figuur 26: Applicatie start	46
Figuur 27: Website register	47
Figuur 28: Install new colony window	47
Figuur 29: Dashboard	48
Figuur 30: Schets dashboard	57

Afkortingenlijst

MOD	Modificatie
JWT	JSON Web Tokens

Inleiding

Dit project heeft als doel het beginproces van MineColonies te vereenvoudigen en meer gebruiksvriendelijk te maken.

MineColonies is een MOD die een gemeenschap probeert na te bootsen in Minecraft. Bij deze MOD is de speler het hoofd van de gemeenschap die ervoor zorgt dat alles in goede banen loopt. Dit proces is in het begin zeer omslachtig, omdat er tussen verschillende gebouwen moet worden gelopen en het overzicht ontbreekt.

Om dit proces vlotter te laten verlopen, is er een webapplicatie gemaakt waarin alle taken en de inventaris van zowel de speler als de gemeenschap duidelijk zichtbaar zijn. Ook is het mogelijk om te beslissen of taken automatisch vervuld worden of dat deze door de speler geselecteerd moeten worden.

Hiervoor is een diepgaand onderzoek nodig naar de verschillende gebruikte MODS, de verschillende frameworks voor de website en de programmeertaal LUA.

Vervolgens moeten er verschillende programma's gemaakt worden, namelijk: de website frontend, de website backend, het C#-programma dat de link vormt tussen Minecraft en de website en de LUA-scripts die de nodige data uit Minecraft extraheert.

Dit rapport focust eerst op Minecraft en de Minecraft MODS die gebruikt worden om de nodige data te extraheren. Daarna focust het rapport op de werking van het C# programma en de connectie met de website en Minecraft.

1 Situering

1.1 Wat is MineColonies?

MineColonies is een MOD voor Minecraft waarbij het doel is om een gemeenschap succesvol te begeleiden en te laten groeien. In deze MOD wordt de speler als hoofd van de gemeenschap gesteld en kan de speler bepalen waar welke gebouwen geplaatst worden. Hierna worden deze gebouwd door de builders van de gemeenschap.



Figuur 1: Gemeenschap in MineColonies

1.2 Wat is het probleem?

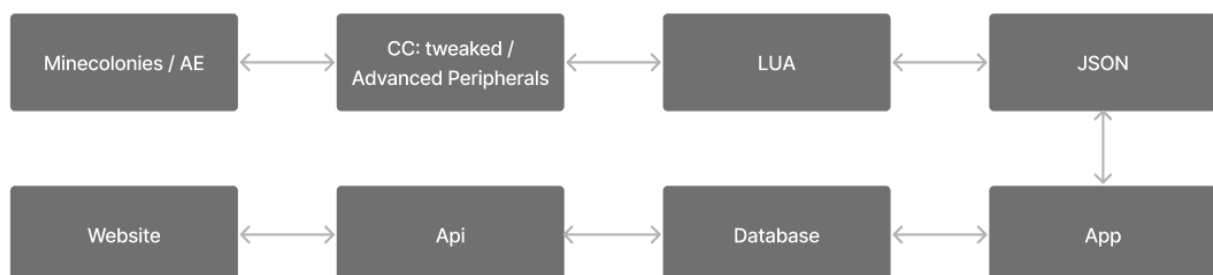
Voor elk gebouw dat gemaakt wordt zijn er veel verschillende resources nodig (Figuur 2). Deze resources moeten door de speler geleverd worden aan de verschillende builders. Dit kan heel snel heel vervelend worden.



Figuur 2: Lijst van resources

1.3 Wat is de oplossing?

Om dit proces vlotter en minder vervelend te maken, is er een website waarop de speler zeer gemakkelijk de verschillende builders en wat zij nodig hebben, kan bekijken. Om dit mogelijk te maken ligt er een link tussen Minecraft en de website. Deze link loopt volgens onderstaande figuur (Figuur 3). Zo blijkt dat CC: Tweaked de data van MineColonies en AE extraheert, waarna deze data via LUA naar een JSON formaat omgezet worden. Deze JSON data wordt dan ingelezen door de App, die op zijn beurt de data doorstuurt naar de database. Hierna kan de Api deze data opvragen van de database en doorsturen naar de website.



Figuur 3: data flowchart

2 Minecraft en MODS

2.1 Overzicht van de gebruikte Minecraft MODS

Om dit project mogelijk te maken, zijn er een aantal MODS nodig naast MineColonies. Hieronder zijn deze kort opgelijst met een korte uitleg erbij.

2.1.1 MineColonies

Deze MOD zorgt voor een simulatie waarbij de speler het stadshoofd is van een gemeenschap die hij in goede banen moet leiden. [1]

Dit is de MOD die wordt gestroomlijnd.

2.1.2 Applied Energistics 2

Applied Energistics 2 is een MOD waarmee spelers items kunnen opslaan in een digitale opslag en deze items kunnen visualiseren op een mooie en begrijpelijke manier. Ook zorgt deze MOD voor het automatisch maken van items eens de speler het systeem dit heeft aangeleerd. [2]



Figuur 4: ME Controller [2]

2.1.3 CC: Tweaked

Deze MOD maakt het mogelijk om te programmeren in Minecraft. Dit gebeurt aan de hand van een speciaal blok genaamd computer (**Error! Reference source not found.** **Error! Reference source not found.**) in het spel waarop scripts geschreven kunnen worden in de LUA-taal. Deze blok kan communiceren met verschillende randapparaten om de basiswerking uit te breiden. [4]



Figuur 5: Advanced Computer [3]

2.1.4 Advanced Peripherals

Dit is een MOD die CC: Tweaked uitbreidt met verschillende extra randapparaten waaronder de Colony Integrator (Figuur 6**Error! Reference source not found.**) en de ME Bridge (Figuur 7**Error! Reference source not found.**). Respectievelijk zorgen deze voor communicatie met MineColonies en Applied Energistics 2. [5]



Figuur 6: Colony Integrator [13]



Figuur 7: ME Bridge [13]

2.2 LUA en Minecraft

CC: Tweaked gebruikt LUA om te interageren met Minecraft.

2.2.1 LUA

LUA is een scripttaal die bekend staat voor zijn snelheid en robuustheid. LUA wordt gebruikt omdat dit de scripttaal is die geïmplementeerd is door CC: Tweaked en Advanced Peripherals. [6]

2.2.2 LUA in CC: Tweaked

LUA in CC: Tweaked is een aangepaste versie van LUA waar een aantal van de basisfuncties van LUA niet aanwezig zijn. Een lijst met welke features geïmplementeerd zijn en welke niet kan gevonden worden op https://tweaked.cc/reference/feature_compat.html [7]

Voor de LUA-scripts is er een object georiënteerde benadering gebruikt. Hiervoor zijn er verschillende LUA-scripts geschreven die met elkaar gelinkt zijn.

```
local function Initialize(monitorWriter)
    -- Cache MonitorWriter
    MonitorWriter = monitorWriter

    -- Check if a savedState exists
    if JsonFileHelper.file_exists("savedState.json") then

        -- Ask the user if they want to use the savedState
        print("previous config file found. Use this? (yes/no)")
        local answer = io.read()
        while not (answer == "yes" or answer == "no") do
            print("Please answer with 'yes' or 'no' (yes/no)")
            answer = io.read()
        end

        -- If the answer is yes then map the peripherals according to the savedState
        if answer == "yes" then
            <Output omitted>
        elseif answer == "no" then
            -- If the answer is no then map the peripherals like new
            MapPeripherals()
        end
    else
        -- If no savedState exists, map the peripherals like new
        MapPeripherals()
    end
    -- Write the config file away
    WriteConfigFile()
end
```

Codefragment 1: Initialize van WrapPeripherals

Als basis voor alles wat met randapparaten te maken heeft, moeten deze randapparaten ingepakt worden zodat er toegang mogelijk is vanuit de code. Hiervoor is er een `wrapPeripherals` (Codefragment 1 **Error! Reference source not found.**) script geschreven dat geïmplementeerd kan worden in andere scripts.

In dit script wordt eerst gekeken of er een `savedState` bestand is. Als dit bestaat en de gebruiker deze configuratie wil, gebruiken dan worden de randapparaten volgens dit configuratiebestand ingesteld. Zo niet, worden eerst alle randapparaten opgelijst, waarna voor elk apparaat gekeken wordt naar het type apparaat en op basis daarvan wordt deze correct opgeslagen.

2.3 Extraheren van takenlijst uit MineColonies

Er zijn 2 soorten taken. De algemene taken en de Buildertaken. De buildertaken bestaan uit alle resources die de builder nodig heeft om een gebouw te maken en de algemene taken bestaan uit alle resources die de Villagers nodig hebben. Binnen de algemene taken staan er soms ook builder taken, maar nooit alle buildertaken die een builder kan hebben. Daarom worden deze taken apart geëxtraheerd van de algemene taken. Bij het extraheren van de algemene taken worden de buildertaken eruit gefilterd.

2.3.1 Extraheren algemene taken

Voor het extraheren van algemene taken worden eerst alle taken geëxtraheerd, hierna worden de taken van de builders eruit gefilterd. (Codefragment 2)

```
local requestData = {}
-- Add the colony name in the data
requestData["Name"] = peripherals.GetColonyIntegrator().getColonyName()
requestData["fingerprint"] = peripherals.GetColonyIntegrator().getColonyID()
-- Add all requests into the data
requestData["Requests"] = peripherals.GetColonyIntegrator().getRequests()
-- Remove all requests that are coming from Builders from the requests table
for i=#requestData["Requests"],1,-1 do
    if not (string.find(requestData["Requests"][i]["target"], "Builder") == nil) then
        table.remove(requestData["Requests"], i)
    end
end
end
```

Codefragment 2: Extraheren algemene taken

Dit wordt gedaan door in de taak naar de target te kijken en als er “Builder” in staat deze vervolgens uit de lijst te verwijderen.

2.3.2 Extraheren buildertaken

Voor het extraheren van de buildertaken moeten eerst de verschillende gebouwen opgehaald worden. Hierna wordt over deze lijst gegaan en als het gebouw een builder is dan worden de specifieke taken van de builder geëxtraheerd. Ook wordt de naam van de builder meegegeven. (Codefragment 3)

```

-- Get all builder requests
local builderRequests = {}
-- Get all buildings
local buildings = peripherals.GetColonyIntegrator().getBuildings()
-- For each building check if it is a builder.
-- if so, get the resources from that builder (this is more then the normal requests above)
for i, building in ipairs(buildings) do
    if not (string.find(building["name"], "builder") == nil) then
        local builder = {}
        -- Add builder name if it has one
        if not (building["citizens"][1] == nil) then
            builder["name"] = building["citizens"][1]["name"]
            monitorWriter.WriteLine("Extracting requests from builder: " .. builder["name"],
peripherals.GetMonitor())
        else
            monitorWriter.WriteLine("Extracting requests from unknown builder.",
peripherals.GetMonitor())
        end
        -- Add builder location
        builder["location"] = building["location"]
        builder["Requests"] =
peripherals.GetColonyIntegrator().getBuilderResources(building["location"])
        table.insert(builderRequests, builder)
    end
end

endend

```

Codefragment 3: extraheren van buildertaken

2.4 Commando's

Commando's zijn altijd bedoeld voor Applied Energistics. Deze commando's kunnen gebruikt worden om automatisch items te verplaatsen of te maken.

Er kunnen twee soorten commando's doorgegeven worden naar Minecraft. Een commando om resources van de playerside storage naar de colonyside te sturen. En een commando om een bepaalde resource te craften in de playerside storage.

2.4.1 Inlezen en uitvoeren van gegeven commando's

Commando's worden door het C# programma gegenereerd en uitgeschreven naar een JSON-formaat. (Codefragment 4)

```
[
  {
    "Item": "minecraft:cobblestone",
    "Amount": 11,
    "NeedsCrafting": false
  },
  {
    "Item": "minecraft:cobblestone_slab",
    "Amount": 16,
    "NeedsCrafting": true
  }
]
```

Codefragment 4: voorbeeld van een commando

Deze commando's worden hierna ingelezen door het LUA-programma en uitgevoerd.

Om deze commando's in te lezen, wordt er eerst gecheckt of de NeedsCrafting variabele true is. Als dit zo is, dan wordt het gevraagde item automatisch gemaakt, anders wordt het item geëxporteerd van het systeem aan de spelerkant naar het systeem aan de colonykant.

Hierna wordt elk uitgevoerd commando verwijderd zodat deze commando's niet meerdere keren na elkaar uitgevoerd worden.

3 C#-programma

3.1 Extractie van colonydata en storedata uit Minecraftfolder

Om de colony- en storagegegevens uit Minecraft te gebruiken, moet deze eerst worden geëxtraheerd uit de Minecraftfolder. Dit proces gebeurt in de data laag, waar alle computers uit de wereldfolder worden gehaald. Vervolgens worden deze computers geclassificeerd als colonycomputers of andere computers.

3.1.1 Colonydata

De colonydata wordt uitgelezen uit requests.json, die zich in een colonycomputer bevindt. De uitgelezen data omvat de naam van de colony, de builderrequests en de requests. Dit proces wordt uitgevoerd met behulp van de ingebouwde functie "JsonSerializer.Deserialize" in DotNet. Aangezien CC: Tweaked een lege array als een object doorstuurt, moeten er eerst enkele vervangingen plaatsvinden voordat de deserialisatie kan worden uitgevoerd. (Codefragment 5)

```
Colonie colonie = null;
string jsonString = System.IO.File.ReadAllText(path + "\\requests.json");
jsonString = jsonString.Replace("\"tags\":{}", "\"tags\":[]");
jsonString = jsonString.Replace("\"tags\": {}", "\"tags\":[]");
jsonString = jsonString.Replace("\"Requests\":{}", "\"Requests\":[]");
jsonString = jsonString.Replace("\"Requests\": {}", "\"Requests\":[]");
jsonString = jsonString.Replace("\"BuilderRequests\":{}", "\"BuilderRequests\":[]");
try
{
    colonie =
    System.Text.Json.JsonSerializer.Deserialize<List<Colonie>>(jsonString)!.[0];
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

Codefragment 5: extraheren van colonydata

3.1.2 storage

De storedata wordt uitgelezen uit aeData.json, die zich in een colonycomputer bevindt. De uitgelezen data omvat een lijst van items van de player, een lijst van items van de colony en een lijst van patterns die beschikbaar zijn. Patterns zijn recipes die automatisch kunnen worden gecraft. Dit proces wordt uitgevoerd met behulp van de ingebouwde functie “JsonSerializer.Deserialize” in DotNet. Aangezien CC: Tweaked een lege array als een object doorstuurt, moeten er eerst enkele vervangingen plaatsvinden voordat de deserialisatie kan worden uitgevoerd. (Codefragment 6)

```
string jsonString = System.IO.File.ReadAllText(path + "\\aeData.json");
jsonString = jsonString.Replace("\"tags\":{}", "\"tags\":[]");
jsonString = jsonString.Replace("\"tags\": {}", "\"tags\":[]");
jsonString = jsonString.Replace("\"colonySide\":{}", "\"colonySide\":[]");
jsonString = jsonString.Replace("\"playerSide\":{}", "\"playerSide\":[]");
jsonString = jsonString.Replace("\"patterns\":{}", "\"patterns\":[]");

try
{
    if (colonie == null) throw new ArgumentException("colonie Does not exist");
    List<ItemsInStorage> Storage =
System.Text.Json.JsonSerializer.Deserialize<List<ItemsInStorage>>(jsonString);
    for (int i = 0; i < Storage.Count; i++)
    {
        colonie.items = Storage[i];
        if (Storage[i].items.colonySide == null)
        {
            colonie.items.items.colonySide = new List<StorageItem>();
        }
        if (Storage[i].items.playerSide == null)
        {
            colonie.items.items.playerSide = new List<StorageItem>();
        }
        if (Storage[i].patterns == null)
        {
            colonie.items.patterns = new List<StorageItem>();
        }
    }
}
```

Codefragment 6: extraheren van storedata

3.2 C# authenticatie

Om als gebruiker in te loggen, moet de gebruiker een e-mailadres en een wachtwoord opgeven. Achter de schermen wordt de gebruiker geauthenticeerd en ontvangt het programma een JWT-

```
HttpContent content = new StringContent("{} ", Encoding.UTF8,
"application/json");
client.DefaultRequestHeaders.Add("Authorization", "bearer " + Token);
client.DefaultRequestHeaders.Add("Accept", "application/json");
HttpResponseMessage response = await client.PostAsync(ApiUrl + "/refresh",
null);
if (response.IsSuccessStatusCode)
{
    // Read the response content as a string
    string responseBody = await response.Content.ReadAsStringAsync();
    JsonNode node = JsonNode.Parse(responseBody);
    string token = node["token"].ToString();
    Token = token;
} else
{
    LoggedIn = false;
}
```

Codefragment 7: ververs refreshtoken

token, waarna er direct een refreshtoken wordt aangevraagd (Codefragment 7 **Error! Reference source not found.**). Deze refreshtoken wordt in elke iteratie van het programma opnieuw aangevraagd en gebruikt in de requests als authorization header. Deze header wordt bij elke iteratie naar de API verstuurd om toegang te verkrijgen.

3.3 Extract recipes uit Minecraft

Als uitbreiding op dit project is er voorzien dat alle recipes van de geselecteerde modpack geëxtraheerd worden. Om dit te doen, wordt er eerst een scan gedaan om alle locaties van de verschillende MODS te ontdekken (Codefragment 8).

```
private void GetModPaths()
{
    if (Path.Exists(InstancePath + "\\mods"))
    {
        ModPaths = Directory.EnumerateFiles(InstancePath +
        "\\mods").ToList();
        var mod = Directory.EnumerateFiles(InstancePath + "\\mods",
        "*AdvancedPeripherals*").ToList();
        if (mod.Count() == 0) throw new Exception("Instance does not have
        AdvancedPeripherals installed");
        mod = Directory.EnumerateFiles(InstancePath + "\\mods",
        "*appliedenergistics2*").ToList();
        if (mod.Count() == 0) throw new Exception("Instance does not have
        Applied Energistics 2 installed");
        mod = Directory.EnumerateFiles(InstancePath + "\\mods", "*cc-
        tweaked*").ToList();
        if (mod.Count() == 0) throw new Exception("Instance does not have
        cc-tweaked installed");
        mod = Directory.EnumerateFiles(InstancePath + "\\mods",
        "*minecolonies*").ToList();
        if (mod.Count() == 0) throw new Exception("Instance does not have
        minecolonies installed");
    }
    else throw new ArgumentException("Instance does not have a mod folder");
    GetMinecraftJarPath();
}
```

Codefragment 8: GetModsPaths

Om de locatie van de Minecraft jar te vinden is wat meer werk nodig. Hiervoor moet eerst bepaald worden of de Instance de Feed The Beast layout gebruikt of de CurseForge layout. Om dit te bepalen, wordt er gekeken naar de layout van de folders. Hierna kan de correcte locatie van de Minecraft jar gevonden worden.

Eens alle jar files gevonden zijn, worden deze files uitgepakt en worden de recipe files gezocht. Hierna worden deze files uitgelezen en afhankelijk van het type recipe worden de juiste gegevens eruit gehaald. Hierbij is er één speciale soort recipe. De “Forge:conditional” recipe. Dit recipe kan meerdere recipes onder een conditional hebben. Om dit toch correct uit te lezen is er gebruik gemaakt van recursie om zo de juiste gegevens te verkrijgen (Codefragment 9).

```
private List<Recipe> ExtractRecipe(JsonNode? recipeNode)
{
    if (recipeNode == null) return new List<Recipe> { };
    List<Recipe> recipes = new List<Recipe>();
    Recipe recipe = new Recipe { Inputs = new List<RecipeItem>(), Results = new
List<RecipeItem>() };
    recipe.Type = recipeNode["type"]!.ToString();
    string resultName = "";
    int resultCount = 0;
    JsonNode? ingredientName = null;
    try
    {
        switch (recipe.Type)
        {
            <other cases omitted>
            case "forge:conditional":
                var jsonRecipes = recipeNode["recipes"]!.AsArray();
                foreach (var jsonRecipe in jsonRecipes)
                {
                    recipes.AddRange(ExtractRecipe(jsonRecipe!["recipe"]));
                }
                break;
        }
    }
    catch (Exception exc)
    {
        Console.WriteLine(exc.ToString());
    }
    return recipes;
}
```

Codefragment 9: Extract recipe recursie

Hierna worden deze Recipes doorgestuurd naar de database.

3.4 Commando's

Commando's dienen als de verbinding tussen C# en Minecraft. Dit wordt bereikt door commando-objecten aan te maken waarin wordt vastgelegd of een item moet worden doorgestuurd of gecraft, en hoeveel ervan nodig is.

Om de commando's aan te maken, wordt eerst gecontroleerd of bepaalde commando's moeten worden gegenereerd door naar de instellingen in de database te kijken voor elke specifieke colony. De drie instellingen die worden uitgelezen zijn: autocrafting, autotools en autoarmor. Als autocrafting voor de colony op false staat, worden alle builderrequests die op autocrafting staan opgevraagd en vergeleken met de geëxtraheerde builderrequests om een autocompleet lijst te maken.

Vervolgens wordt deze lijst doorlopen om de commando's aan te maken. Hierbij wordt eerst gecontroleerd of er voldoende items in de colonyopslag aanwezig zijn, zodat deze niet meer

worden omgezet naar commando's. Hierna wordt er gekeken of er genoeg items zijn in de playerstorage. Als dit het geval is, wordt er een commando aangemaakt met NeedsCrafting op false. Als er niet genoeg items zijn, wordt er gecontroleerd of er een pattern bestaat. Indien dit het geval is, wordt er een commando aangemaakt met NeedsCrafting op true.

4 Websiteontwikkeling

4.1 Overzicht van het Ontwikkelingsproces

Het proces van het ontwikkelen van de website begint met het maken van ontwerpen en een flowchart in Figma. Na het kiezen van een kleurenpalet en het beslissen over de technologieën, wordt de website eerst gebouwd met Javascript, HTML en SCSS. Vanwege de behoefte aan een meer robuuste oplossing, wordt de website vervolgens herbouwd met het Nuxt framework, wat enige uitdagingen met zich meebrengt. Voor de interactieve gebruikershandleiding wordt Three.js gekozen om een 3D-ervaring te creëren, waarbij een interactieve Minecraft Steve wordt geïmplementeerd. Uiteindelijk wordt alle data van de API geïntegreerd en netjes weergegeven op het dashboard. Hieronder volgen de gedetailleerde hoofdstukken voor elk onderdeel van dit proces.

4.2 Ontwerp met Figma

4.2.1 Flowchart en doelen

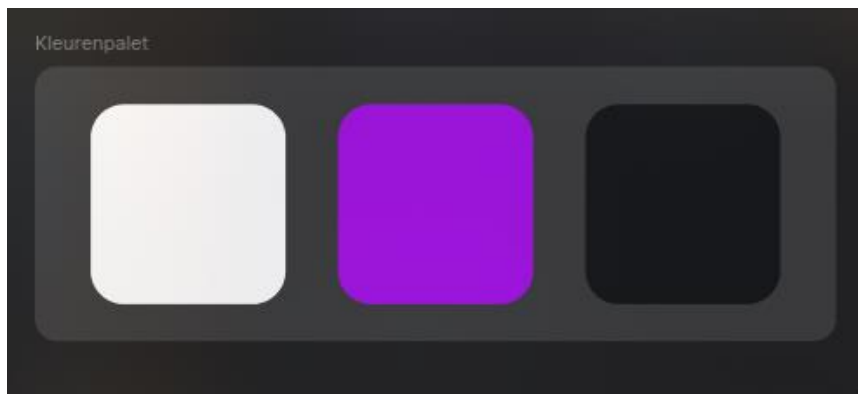
Als eerste stap wordt samengezeten met het team om de doelen vast te stellen en een flowchart te maken. Hierdoor kan verder gewerkt worden aan het design en is het duidelijk wat uiteindelijk op de website te zien moet zijn.



Figuur 8: Figma flowchart website

4.2.2 Eerste Figma schetsen & kleurenpalet selectie

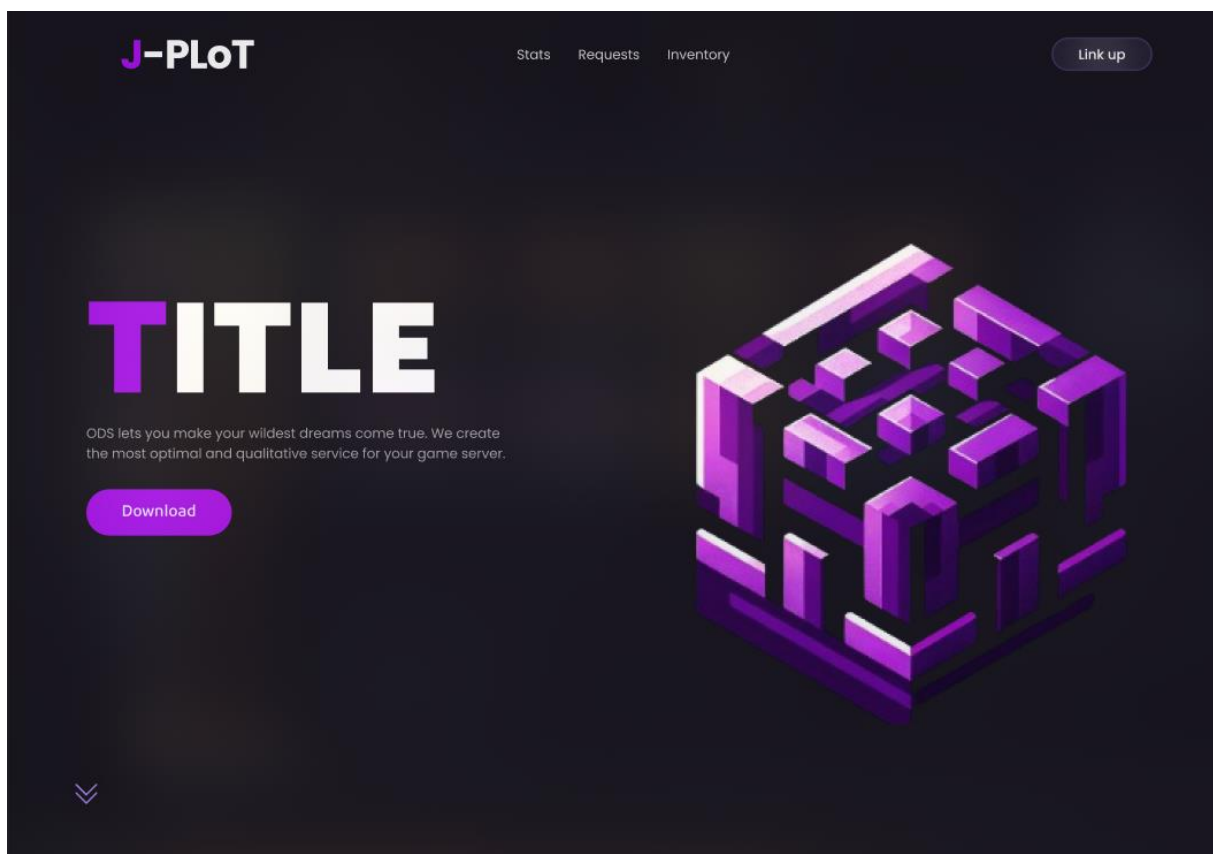
De tweede stap in het ontwerpproces is het maken van ruwe schetsen om de basisstructuur van de website te bepalen. Er wordt geëxperimenteerd met verschillende kleurencombinaties om een samenhangend en visueel aantrekkelijk kleurenpalet te kiezen, dat dan doorheen de hele website wordt geïmplementeerd. Hierbij worden kleuren uitgekozen die zowel esthetisch aantrekkelijk als functioneel zijn voor de gebruikerservaring. (Figuur 9)



Figuur 9: Kleurenpalet

4.2.3 Feedback en revisie

Na het maken van de eerste ontwerpen worden deze gedeeld met het team voor feedback en worden er verbeteringen aangebracht. (Figuur 10)

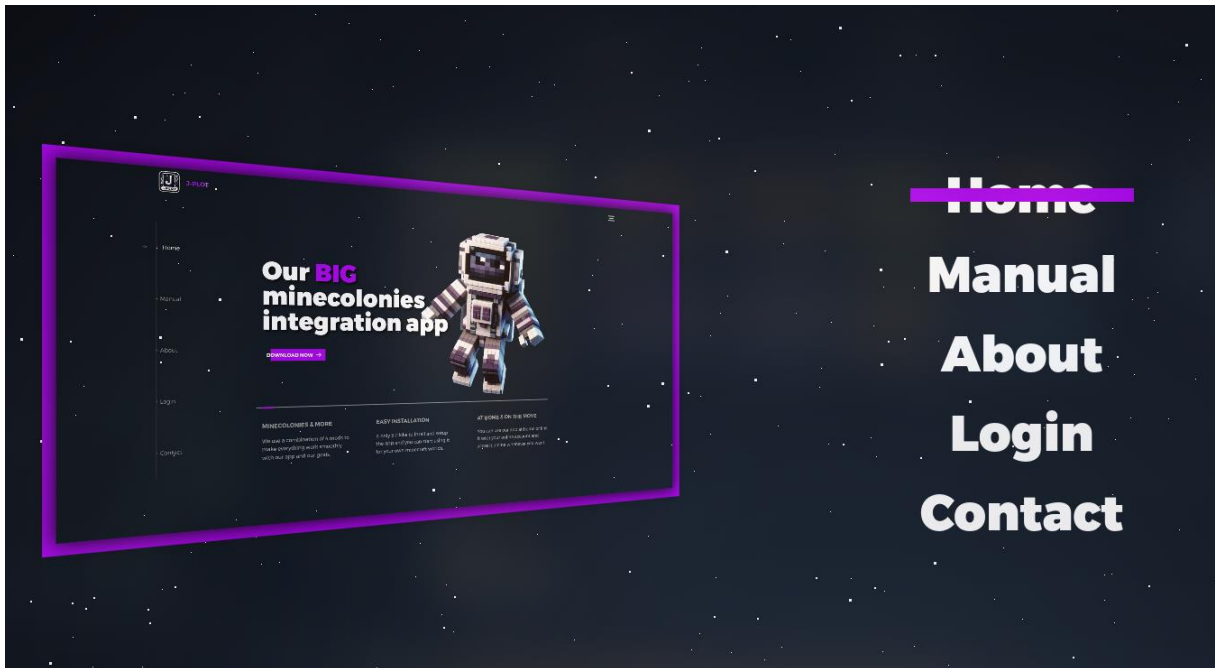


Figuur 10: Figma design na feedback

4.3 Technologiekeuze: Javascript, HTML en SCSS

4.3.1 Voordelen van Javascript

JavaScript wordt gekozen voor de interactieve elementen en dynamische functionaliteiten. Het biedt de flexibiliteit om complexe interacties en real-time updates op de webpagina's te implementeren. Dit wordt gebruikt om de website scrollable te maken en een uniek burger menu te implementeren (Figuur 11).



Figuur 11: Burger menu

4.3.2 Structuur met HTML

HTML wordt gebruikt voor de basisstructuur van de webpagina's. Het zorgt voor een semantische opbouw van de inhoud, wat essentieel is voor zowel toegankelijkheid als zoekmachineoptimalisatie (Codefragment 10).

```
<!DOCTYPE html>

<html lang="en">

<head>

  <title>J-PLOT</title>

  <link rel="icon" href=assets/img/logo3.png>

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="description" content="HTML5 website template">

  <meta name="keywords" content="global, template, html, sass, jquery">

  <link rel="stylesheet" href="assets/css/main.sass">

</head>

<body...>

</html>
```

Codefragment 10: HTML-structuur voor zoekmachine optimalisatie

4.3.3 Styling met SCSS

SCSS wordt gebruikt voor een gestructureerde en herbruikbare CSS. Het biedt de mogelijkheid om gebruik te maken van variabelen, geneste regels en mixins, wat de ontwikkeling van onderhoudbare en schaalbare stijlen vergemakkelijkt. In het project worden de variabelen (Codefragment 11) gebruikt om het eerder samengestelde kleurenpalet doorheen de hele website vlot en efficiënt toe te passen.

```
// Colors

$white: #fff

$black: #0c0c0c

$highlight: #ad04f0

$muted-gray: #a1a1a1
```

Codefragment 11: _vars.sass file met variabelen

4.4 Overstap naar Nuxt framework

4.4.1 Wat is Nuxt?

Nuxt is een gratis en open source framework gebaseerd op Vue.js, Nitro en Vite dat wordt gebruikt voor het bouwen van server-side gerendeerde applicaties en statische websites. Het biedt veel voordelen, zoals automatische code-splitting, een krachtige modulebibliotheek en een heel uitgewerkte, efficiënte ingebouwde debugging tool.

4.4.2 Installatie en configuratie van Nuxt

Het opzetten van de juiste configuratie voor de projectomgeving met Nuxt gebeurt met behulp van de Nuxt documentatie [8].

Dit kan doormiddel van een paar simpele commando's:

Open een terminal (als Visual Studio Code wordt gebruikt, kan een geïntegreerde terminal worden geopend) en gebruik het volgende commando om een nieuw starterproject te creëren:

```
| pnpm dlx nuxi@latest init <project-name>
```

Open de projectmap in Visual Studio Code:

```
| code <project-name>
```

Of navigeer naar de projectmap vanuit de terminal:

```
| cd <project-name>
```

4.4.3 Uitdagingen en oplossingen

Tijdens de overgang naar Nuxt verschijnen verschillende problemen, zoals compatibiliteitsproblemen en prestatieoptimalisaties. Deze worden opgelost door grondig onderzoek, het raadplegen van documentatie en het implementeren van best practices.

Het overzetten van de bestaande JavaScript naar het Nuxt framework wordt opgelost door sommige functionaliteiten om te vormen of anders aan te pakken. Bijvoorbeeld de unieke burger menu en de login logica.

De SCSS wordt niet goed overgebracht, waardoor de styling niet consistent wordt toegepast. Dit wordt opgelost door sass te gebruiken en alles naar één main te importeren en dit bestand toe te voegen in de Nuxt config en sass te installeren in het project met dit commando in de terminal:

```
| pnpm install -D sass
```

4.5 Gebruik van Three.js

4.5.1 Wat is Three.js?

Three.js is een Javascript-bibliotheek voor het maken van 3D-computergraphics in een webbrowser. Het biedt een eenvoudige manier om complexe 3D-modellen en animaties te maken en te renderen.

4.5.2 Leren en implementeren van Three.js

Door gebruik te maken van de Three.js documentatie [9] en online voorbeelden [10] wordt het mogelijk om de basisprincipes van 3D-programmeren onder de knie te krijgen. JavaScript wordt gebruikt om dit te implementeren in de HTML-pagina (Codefragment 12).

```
<script type="module" src="main.js"></script>
```

Codefragment 12: Javascript implementatie

In dit JavaScript bestand wordt dan de Three.js bibliotheek geïmporteerd (Codefragment 13).

```
import * as THREE from 'three';
```

Codefragment 13: Javascript import statement

4.5.3 Interactieve Minecraft Steve

Het creëren van een interactieve Minecraft Steve die op de website/manual kan rondwandellen en interactie heeft met knoppen en blokken die zich op de website bevinden. Dit kan ook uitgezet worden voor toegankelijkheid.

4.5.4 Integratie met Nuxt

Het integreren van de Three.js-elementen binnen het Nuxt framework verloopt vlot, voornamelijk omdat er al rekening mee gehouden wordt tijdens het proces van de overstap naar het Nuxt framework.

Het JavaScript-bestand wordt geïmporteerd in het Nuxt-project, hierna wordt dit script met alle logica in de handleidingpagina met een simpele lijn code toegevoegd.

4.6 Data-integratie en dashboard ontwikkeling

4.6.1 Koppeling met de API

Het opzetten van de verbinding met de API en het ophalen van de benodigde data gebeurt met asynchrone fetches in de scripts van de pagina's waar deze specifieke data nodig is..

4.6.2 Weergave van data op het dashboard

Het presenteren van de data op een georganiseerde en gebruiksvriendelijke manier op het dashboard gebeurt met verschillende visuele elementen zoals knoppen, tabellen en modals. De opgehaalde data wordt dan getoond op de website via deze code in de HTML (Codefragment 14).

```
<div class="col-span-4 md:col-span-1 p-4 border border-gray-300 rounded">
  <h2 class="text-xl font-semibold mb-2">Select World</h2>
  <select v-model="selectedWorld" @change="fetchColonies" class="w-full p-2 border
border-gray-300 rounded">
    <option value="" disabled>Select a world</option>
    <option v-for="world in worlds" :key="world.worldId" :value="world.worldId">{{
world.name }}</option>
  </select>
</div>
```

Codefragment 14: data weergave van de opgehaalde werelden

4.6.3 Data-ophaalmethode en voorbeeldcode

Gebruik van asynchrone JavaScript-methoden om data op te halen en te verwerken zorgt voor een efficiënte en responsieve gebruikerservaring.

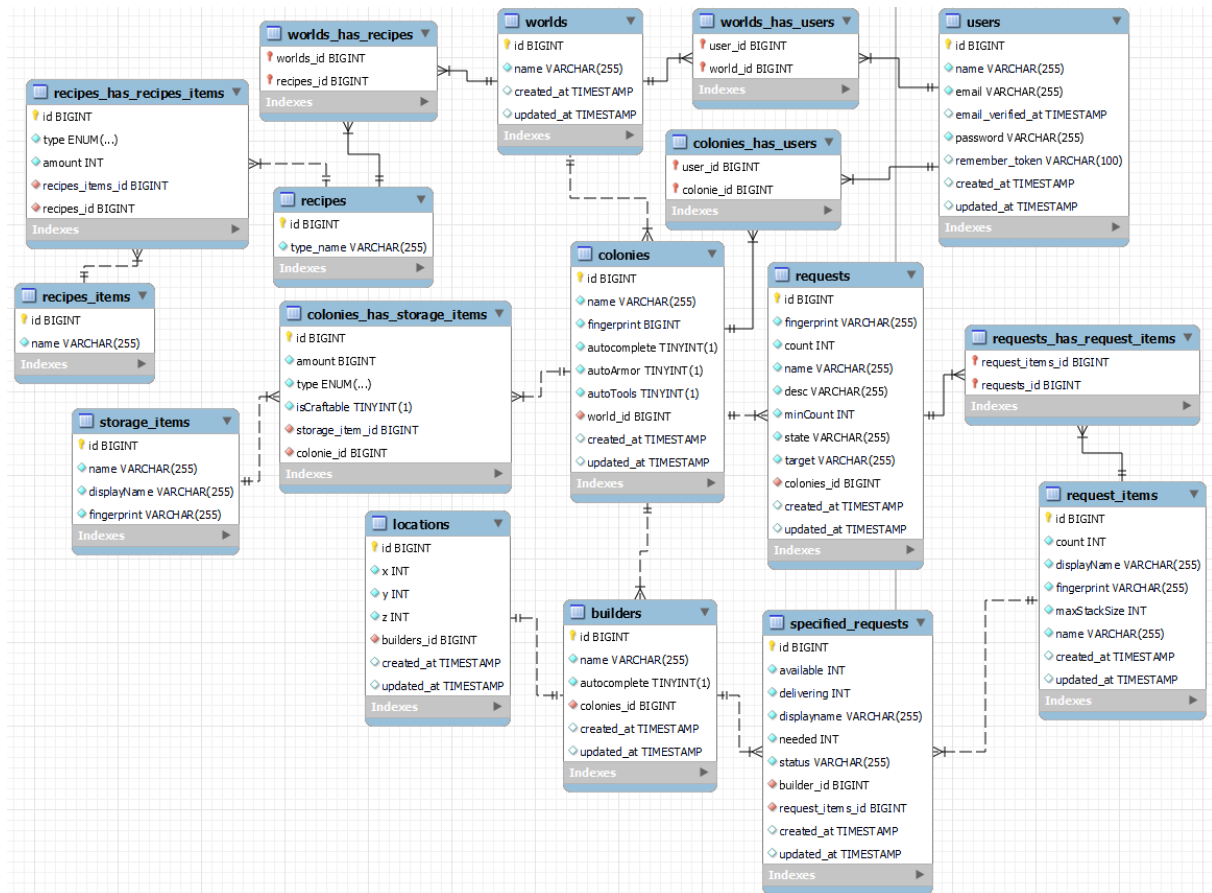
Zoals u ziet in codefragment (Codefragment 15) wordt er een fetch gedaan om de werelden van de API op te halen. De data die ontvangen wordt, wordt gereturned tenzij er foutieve of geen data wordt ontvangen. Ook wordt de `loading.value` op `false` gezet zodat de ingebouwde laadfunctie stopt met het laadteken te vertonen.

```
const fetchWorlds = async () => {
  try {
    const response = await
    axios.get('https://minecraftapi.thibeprovoost.ikdoeict.be/api/worlds')
    worlds.value = response.data
  } catch (error) {
    console.error('Error fetching worlds:', error)
  } finally {
    loading.value = false
  }
}
```

Codefragment 15: Asynchrone fetch van de werelden

4.7 Backendontwikkeling

Voor de backend wordt gebruik gemaakt van Laravel met eloquent models. Om een goede API te schrijven is er een duidelijk databankontwerp (Figuur 12) gemaakt.



Figuur 12: Databank ontwerp

4.7.1 Beveiliging

Om te voorkomen dat users aan andermans werelden kunnen, is er beveiliging op de meeste endpoints. Dit aan de hand van JWT-tokens: deze autoriseren de gebruiker en zorgen ervoor dat de gebruiker enkel zijn data terugkrijgt. Alle routes hebben de middleware behalve de routes login en register.

Er werd nog niet gewerkt met rolgebaseerde authenticatie door het gebrek aan tijd. Een uitbreiding op het project zou multiplayer ondersteuning zijn, zodat gebruikers samen een colony kunnen beheren. De databank is hier al deels op voorzien, al zijn sommige zaken nog niet ondersteund.

4.7.2 Laravel problemen

Er zijn tijdens de ontwikkeling van het project meerdere problemen vastgesteld. Endpoints deden soms iets anders dan wat er verwacht werd waardoor er soms verwarring ontstond. Ook tijdens het uitrollen zijn er fouten waargenomen. Tijdens de eerste test werd duidelijk dat de API een limit heeft op het aantal requests dat deze kan ontvangen.

Om dit op te lossen zijn er verschillende mogelijkheden op tafel gelegd. De limit proberen verhogen, requests optimaliseren of de API op een eigen server plaatsen. Na alle opties te

proberen werd er besloten om de API op een eigen server te plaatsen. Hierdoor is het probleem van de limit op requests opgelost.

5 Werking API

Om de gegevens van Minecraft efficiënt te kunnen verwerken, wordt gebruik gemaakt van een Web API waarnaar er requests worden gestuurd vanuit de C# applicatie. Er wordt een onderscheid gemaakt tussen POST, GET, PUT, DELETE requests.

5.1 POST requests

5.1.1 Werelden

Werelden worden aangemaakt in de databank op het endpoint: “**/api/worlds**”. Om een wereld aan te maken, dient in de requestbody de naam van de wereld meegegeven te worden (Codefragment 16). Deze naam is de locatie van de world op de harde schijf van de gebruiker, waarbij de “/” naar “-” zijn veranderd. Dit is een unieke waarde om de world gemakkelijk te kunnen selecteren.

```
{
  "name": "C:-Users-Testuser-curseforge-minecraft-Instances-MineColonies
integration-saves-test world"
}
```

Codefragment 16: POST Request body wereld

Na een succesvolle POST request en nadat de wereld wordt aangemaakt, wordt er een statuscode 201 teruggestuurd. In het geval dat de wereld al bestaat, wordt een statuscode 400 teruggestuurd.

5.1.2 Colonies

Nadat de werelden zijn aangemaakt in de databank, kunnen er ook colonies in de databank op het endpoint: “**/api/colonies**” aangemaakt worden. Om een colony te maken dient in de requestbody de naam en de worldid meegegeven te worden (Codefragment 17).

```
{
  "name": "ikecolonie",
  "world_id": 2
}
```

Codefragment 17: POST Request body colony

Na een succesvolle POST request en nadat de colony wordt aangemaakt, wordt er een statuscode 201 teruggestuurd. In het geval dat de colony al bestond wordt een statuscode 400 teruggestuurd.

5.1.3 Requests

Nadat er een colony aangemaakt is, kunnen er requests op het endpoint, **“/api/colonies/requests”**, worden aangemaakt en opgeslagen in de databank. De endpoint verwacht een lijst van requestobjecten. Een requestobject heeft naast de data van een request ook de naam van de colony en de worldid om deze requests toe te voegen aan de databank (Codefragment 18).

```
[
  {
    "colonies_id": 4,
    "fingerprint": "47e648c7-09b6-4f24-b497-0ea07d19e398",
    "count": 1,
    "name": "Chestplate",
    "desc": "Chestplate with minimal level: Leather and with maximal level:
Gold",
    "id": "47e648c7-09b6-4f24-b497-0ea07d19e398",
    "minCount": 1,
    "state": "IN_PROGRESS",
    "target": "Archer Case B. McGee",
    "items": [
      {
        "count": 1,
        "displayName": "[Golden Chestplate]",
        "fingerprint": "E969C47F2E35EC0378D7D0D55EE412C4",
        "maxStackSize": 1,
        "name": "minecraft:golden_chestplate"
        "nbt": {
          "Damage": {
            "ValueKind": 4
          }
        }
      }
    ]
  }
]
```

Codefragment 18: POST Request body request

5.1.4 Builderrequests

Nadat er een colonie aangemaakt is kunnen er builders met hun requests op het endpoint, **`/api/colonies/builderrequests`**, aangemaakt worden. De endpoint verwacht een lijst van builderrequestsobjecten. Een builderrequestsobject heeft naast de data van een builderrequests, een colony-id om deze builderrequests toe te voegen aan de databank (Codefragment 19).

```
[
  {
    "colonies_id": 5,
    "location": {
      "x": 359,
      "y": 67,
      "z": -191
    },
    "name": "Frankie T. Claybrook",
    "Requests": [
      {
        "available": 0,
        "delivering": 0,
        "displayName": "Cobblestone",
        "item": {
          "count": 1,
          "displayName": "[Cobblestone]",
          "fingerprint": "34C0EF2C2CD360FB88A0AD2798E4354D",
          "maxStackSize": 64,
          "name": "minecraft:cobblestone",
          "nbt": {},
          "tags": [
            "minecraft:item/minicolonies:reduceable_product_excluded",
            "minecraft:item/minicolonies:blacksmith_product_excluded"
          ]
        },
        "needed": 11,
        "status": "DONT_HAVE"
      }
    ]
  }
]
```

Codefragment 19: POST Request body builder requests

5.1.5 Storage items

Een colony heeft ook items nodig om requests uit te voeren. Deze items worden ook opgeslagen in de databank. Items van een colony worden opgeslaan op het endpoint “*/api/storage_items*”, de requestbody is een array van storage items (Codefragment 20**Error! Reference source not**

```
[
  {
    "name": "minecraft:birch_log",
    "displayName": "Birch Log",
    "amount": 2147483647,
    "fingerprint": "451A63978FC37B40F5357A1B313A4C56",
    "isCraftable": false,
    "type": "0",
    "colonie_id": 1
  },
  {
    "name": "minecraft:cherry_log",
    "displayName": "Cherry Log",
    "amount": 2147483647,
    "fingerprint": "DB7694AAF78AF9F1C87A2FD701D3A13B",
    "isCraftable": false,
    "type": "0",
    "colonie_id": 1
  }
]
```

Codefragment 20: POST Request body Storage items

found.).

5.1.6 Recipes items

Een wereld heeft ook verschillende manieren om items te maken. Om een item te maken, is er een recipe nodig. Deze wordt aangemaakt en bijgehouden in de databank door naar het endpoint “**/api/recipes**” een request te doen. In de requestbody(Codefragment 21) zit er een array van world id. Dit zijn alle werelden waarbij de recipes worden aangemaakt. Heeft een wereld al recipes dan gebeurt er niets.

```
{
  "worlds": [
    1,
    2
  ],
  "recipes": [
    {
      "Results": [
        {
          "Items": [
            "minecraft:iron_block"
          ],
          "Amount": 1
        }
      ],
      "Inputs": [
        {
          "Items": [
            "forge:ingots/iron"
          ],
          "Amount": 9
        }
      ],
      "Type": "minecraft:crafting_shaped"
    }
  ]
}
```

Codefragment 21: POST Request body recipe items

5.2 GET requests

5.2.1 Werelden

Werelden kunnen worden opgevraagd op het endpoint “**/api/worlds**”. Dan krijgt de huidige ingelogde user al zijn werelden terug. Als returnbody krijgt het programma een JSON-object terug (Codefragment 22).

```
{
  "data": {
    "id": 4,
    "name": "C:-Users-Jonas-curseforge-minecraft-Instances-MineColonies
integration-saves-test world v2",
    "created_at": "2024-05-07T09:10:10.000000Z",
    "updated_at": "2024-05-07T09:10:10.000000Z",
    "colonies": [
      {
        "id": 4,
        "name": "test",
        "autocomplete": false,
        "autoArmor": false,
        "autoTools": false
      }
    ]
  }
}
```

Codefragment 22: Response body werelden

5.2.2 Colonies

Colonies kunnen worden opgevraagd op het endpoint “*/api/colonies*”. Dan krijgt de huidige ingelogde user al zijn colonies terug met alle builders en requests. Als returnbody krijgt het programma een JSON-object terug (Codefragment 23).

```
{
  "data": {
    "id": 4,
    "name": "test",
    "autocomplete": false,
    "autoArmor": false,
    "autoTools": false,
    "builderRequests": [
      {
        "id": 1,
        "name": "jeff",
        "autocomplete": true,
        "location": {
          "x": 1,
          "y": 2,
          "z": 54
        },
        "colonies_id": 4,
        "created_at": "time"
      }
    ]
  }
}
```

Codefragment 23: Response body colonies

5.2.3 Recipes

Recipes kunnen worden opgevraagd per wereld op het endpoint “**/api/recipes/:id**” waarbij id het worldid is. Dan krijgt de huidige ingelogde user al zijn colonies terug met alle builders en requests. Als returnbody krijgt het programma een JSON-object terug (Codefragment 24).

```
{
  "data": {
    "id": 1,
    "name": "AdminsWorld",
    "created_at": "2024-05-30T11:49:32.000000Z",
    "updated_at": null,
    "recipies": [
      {
        "id": 1,
        "type": "minecraft:crafting_shaped",
        "items": {
          "results": [
            {
              "id": 1,
              "name": "minecraft:iron_block",
              "amount": 1,
              "type": "Result"
            }
          ],
          "inputs": [
            {
              "id": 2,
              "name": "forge:ingots/iron",
              "amount": 9,
              "type": "Input"
            }
          ]
        }
      }
    ]
  }
}
```

Codefragment 24: Response body Recipes

5.3 PUT requests

Colonies en builders hebben parameters om de autocomplete van hun taken aan of uit te zetten. Om deze waarden te kunnen updaten wordt er een PUT request naar de endpoints gestuurd: ***“/api/colonies/:id”*** en ***“/api/builderrequests/:id”***. Hierbij is id de unieke id van de colony of builder. In de requestbody wordt de parameter die moet worden getoggled meegegeven (Codefragment 25).

```
{  
  "autocomplete": false  
}
```

Codefragment 25: PUT requests

5.4 DELETE requests

Er is maar 1 DELETE request, dit is het endpoint ***“/api/colonies/:id”***. Id is hier de colony-id van de colonie dat verwijderd moet worden.

Conclusie

Aan de hand van een API, een website, een applicatie en meerdere scripts, is het voor de gebruiker veel overzichtelijker om de verschillende buildertaken te kunnen bekijken en eventueel automatisch te voltooien.

Voor de API bestaat een uitgebreide database waarin colonies opgeslagen kunnen worden. Ook is er een authenticatie systeem gebaseerd op JWT-tokens en endpoints waarnaar de gebruiker requests kan sturen.

De scripts bestaan uit een reeks LUA-scripts die allemaal met elkaar samenwerken om de benodigde data in Minecraft om te zetten naar JSON-files zodat deze kunnen ingelezen worden door de applicatie.

De applicatie kan de data over een colony uit minecraft halen. Hiervoor gebruikt het de files die zijn aangemaakt door de lua scripts. Deze data wordt dan verwerkt in het programma tot objecten die via API requests naar de databank worden gestuurd. Ook zal het de database queriën om de automatisch te voltooien requests om te zetten naar commando's die naar minecraft worden gestuurd.

Het ontwikkelingsproces van de website leidt tot een gebruiksvriendelijke en visueel aantrekkelijke webapplicatie voor MineColonies. Het vaststellen van doelen en het maken van een flowchart zorgt voor een solide basis. De keuze voor JavaScript, HTML en SCSS resulteert in een dynamische en interactieve website. De overstap naar het Nuxt framework brengt uitdagingen, maar deze worden overwonnen door diepgaand onderzoek en aanpassingen.

Three.js integreert succesvol met Nuxt, waardoor Minecraft Steve in 3D over de website kan wandelen en interacties kan hebben met de omgeving. De API zorgt voor soepele data-integratie, waardoor real-time gegevens efficiënt op het dashboard worden weergegeven.

De website biedt nu een robuuste en schaalbare oplossing, klaar voor toekomstige uitbreidingen. Het overzichtelijke ontwerp en de automatisering van taken verbeteren het gebruiksgemak en de efficiëntie voor spelers van MineColonies.

Handleiding

Minecraft:

Creëer een modpack met de benodigde mods: MineColonies, CC: Tweaked, Advanced Peripherals en Applied Energistics.

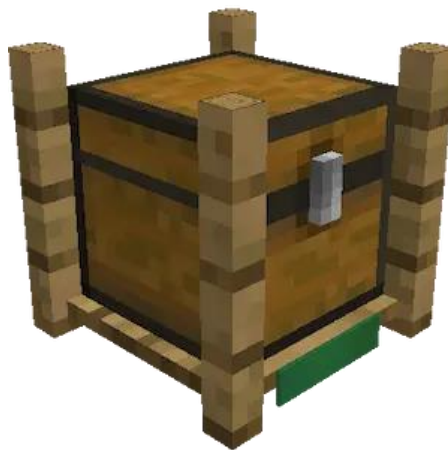
Start de modpack op en maak een wereld aan.

Start in deze wereld een Colony

Maak een warehouse (Figuur 13) en een courier (Figuur 14).



Figuur 13: Warehouse [11]



Figuur 14: Courier [11]

Maak 2 ME systems aan, 1 voor de items van de colony, 1 voor de items van de speler.

Verbind de warehouse met Applied Energistics met een ME Storage bus (Figuur 15) met de colony ME system.

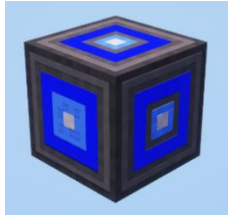


Figuur 15: Warehouse met storage bus

Verbind een Advanced computer (Figuur) met 2 Me Bridges (Figuur 19), 2 ME Interfaces (Figuur 18), 1 Colony integrator (Figuur 17) en 1 monitor (Figuur 16) zoals in onderstaande foto's (Figuur 21 en Figuur 22). Houd hierbij goed bij welke ME bridge en ME interface met welk ME-system verbonden is.



Figuur 20:
Advanced
computer [3]



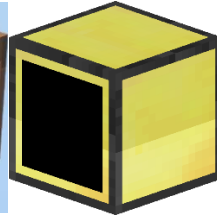
Figuur 19: ME-
Bridge [13]



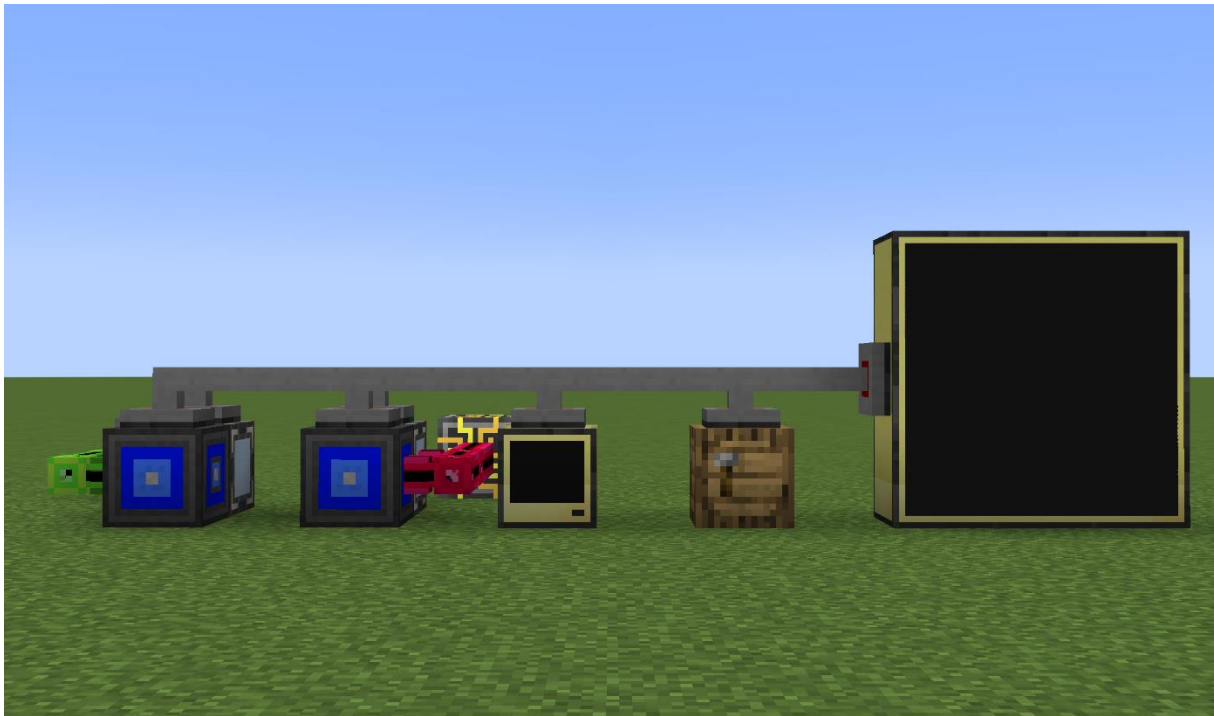
Figuur 18: ME
interface [14]



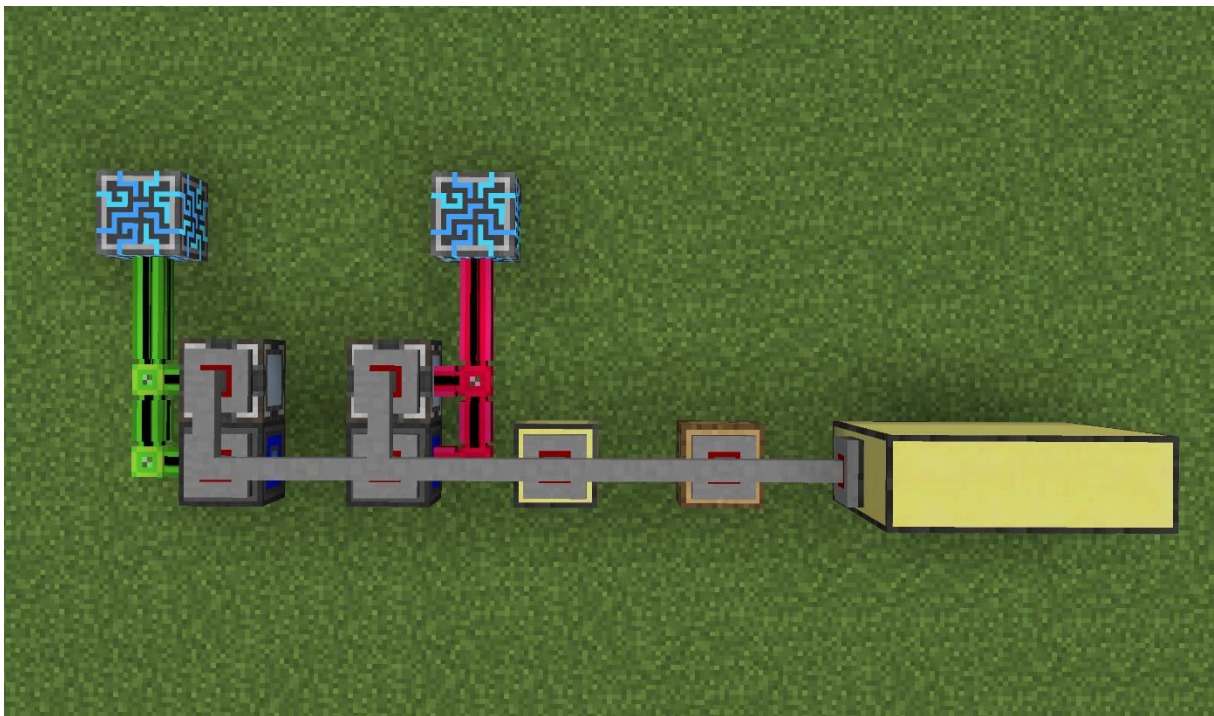
Figuur 17: colony
integrator [13]



Figuur 16: Advanced
monitor [3]



Figuur 21: Setup vooraanzicht



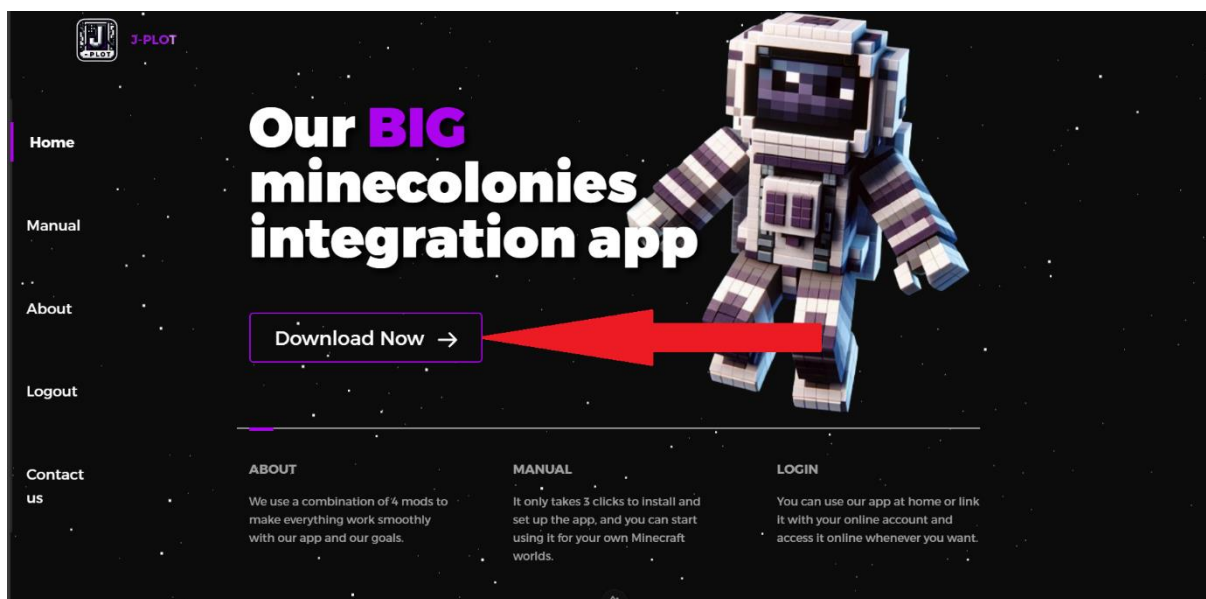
Figuur 22: Setup bovenaanzicht

Open hierna de Advanced computer en voer het volgende commando uit: Edit lua.lua

Druk hierna op control en daarna op enter als save geselecteerd is.

Programma:

Surf naar <https://j-plot.lucavandenweghe.ikdoeicme-systemhttps://j-plot.lucavandenweghe.ikdoeict.be/>





Figuur 23: Website home page

Klik op de Download Now knop (Figuur 23)

Klik op de Download knop. Dit downloadt een zipfile.

Unzip de zipfile.

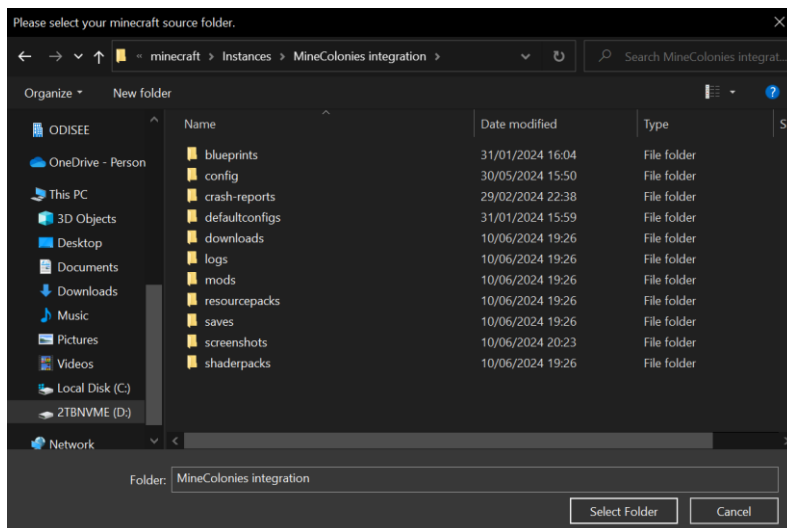
Run de MinecoloniesInstaller.msi file (Figuur 24).

Name	Date modified	Type	Size
 cab1.cab	10/06/2024 18:52	Cabinet File	1.043 KB
 MinecoloniesInstaller.msi	10/06/2024 18:52	Windows Installer Pa...	772 KB

Figuur 24: Unzipped folder

Run het programma

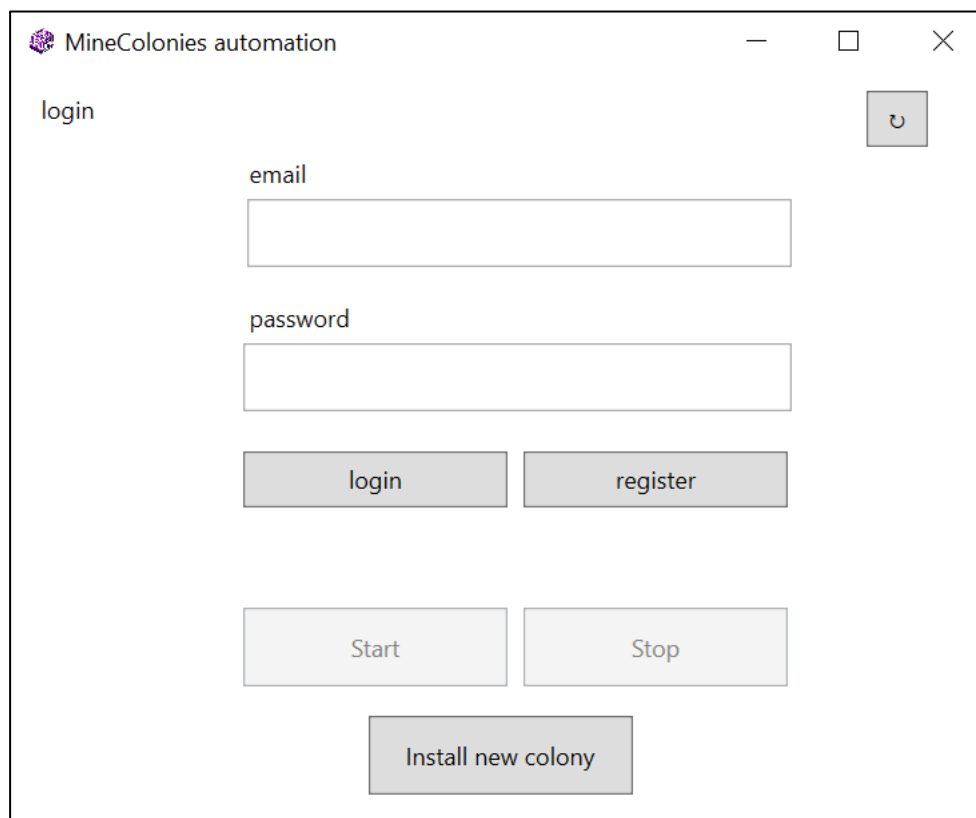
Als dit de eerste keer is dat u het programma runt dan zal volgend venster opengaan (Figuur 25).



Figuur 25: Instance selectie

Selecteer hierin het pad naar uw modpack instance.

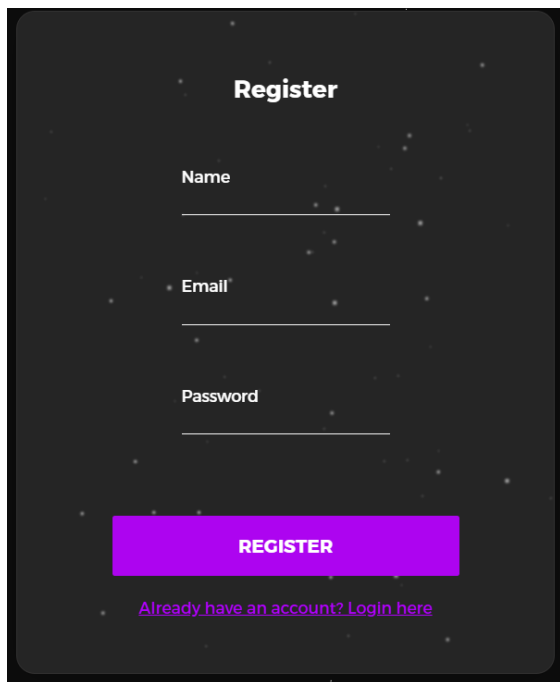
Hierna heeft u een aantal opties: Login, Register (Figuur 26) en Install new colony (Figuur 26).



Figuur 26: Applicatie start

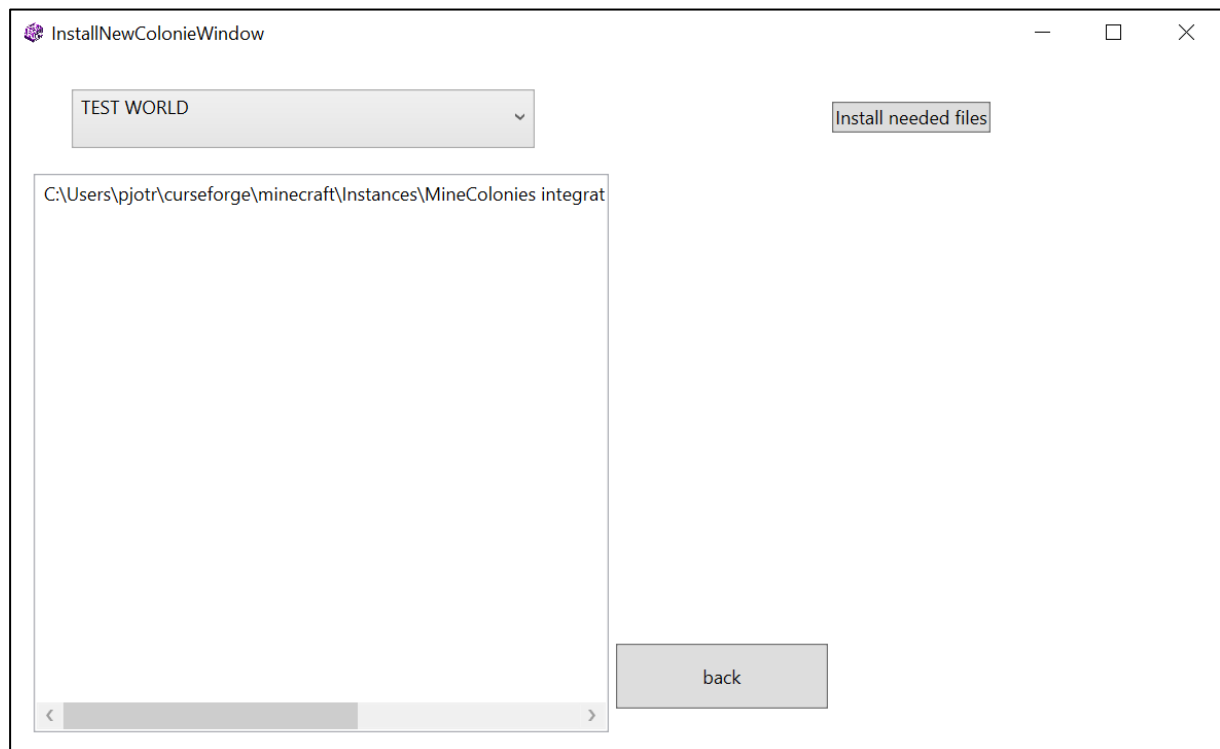
Login: Met deze knop kan u inloggen met hetzelfde account als op de website. Als u geen account hebt kunt u een aanmaken via de registerknop.

Register: Deze knop stuurt u door naar de website waar u een account kunt aanmaken (Figuur 27).



Figuur 27: Website register

Install new colony: Op dit venster kan u de wereld selecteren waarin u een colony wilt installeren. Als er in deze wereld een correcte configuratie bestaat dan zal deze in de lijst staan. Selecteer de computer waarin u de juiste bestanden wilt installeren, en klik de Install needed files knop ().



Figuur 28: Install new colony window

Hierna kunt u op de back knop drukken.

Als u dit alles gedaan hebt kunt u op de Start knop drukken om de synchronisatie te starten.

Nu runt u op de computer in Minecraft volgend commando: main

Volg de stappen op de terminal.

Nu kunt u alles op de website zien als u op de website inlogt (Figuur 29).

Dashboard

Select World

Select Colony

Selected World: C:-Users-Jonas-curseforge-minecraft-Instances-MineColonies Integration-saves-test world v2

Selected Colony: DemoColony

Autocomplete All Requests

Autocomplete Tools

Autocomplete Armor

Requests

ID	Name	Description	Target	State	Count	Min Count	Created At
5	1-16 Dandelion	1-16 Dandelion	Doctor Mackenzie B. Alessi	IN_PROGRESS	16	1	28/05/2024, 10:37:29
6	Shield	Shield with minimal level: Wood or Gold and with maximal level: Wood or Gold	Knight Bellamy B. Pericherla	IN_PROGRESS	1	1	28/05/2024, 10:37:29
7	1-16 Poppy	1-16 Poppy	Doctor Mackenzie B. Alessi	IN_PROGRESS	16	1	28/05/2024, 10:37:29

Builder Requests

ID	Name	Autocomplete	Created At	Actions
7	Joan P. Crawley	False	28/05/2024, 10:37:29	More Info
8	Anderson X. Hampden	False	28/05/2024, 10:37:29	More Info
9	Kabir N. Baynton	False	28/05/2024, 10:37:29	More Info

Figuur 29: Dashboard

Literatuurlijst

- [1] W. Mark Fisher, „wiki.minecolonies.ldtteam,” MineColonies, 1 1 2021. [Online]. Available: <https://wiki.minecolonies.ldtteam.com/>. [Geopend 25 03 2024].
- [2] „<https://guide.appliedenergistics.org/1.20.1/index>,” Applied Energistics 2, 1 1 2024. [Online]. Available: <https://guide.appliedenergistics.org/1.20.1/index>. [Geopend 25 03 2024].
- [3] Feed The Beast, „ftbwiki.org,” [Online]. Available: https://ftbwiki.org/File:Block_Advanced_Computer.png. [Geopend 19 March 2024].
- [4] „tweaked.cc,” tweaked.cc, 25 03 2024. [Online]. Available: <https://tweaked.cc/>. [Geopend 24 03 2024].
- [5] „advanced-peripherals.de,” SyntheticDev, [Online]. Available: <https://docs.advanced-peripherals.de/>. [Geopend 25 03 2024].
- [6] D. o. C. S. o. PUC-Rio, „lua.org,” Lua.org, 18 02 2024. [Online]. Available: <https://www.lua.org/about.html>. [Geopend 25 03 2024].
- [7] cc: Tweaked, [Online]. Available: <https://tweaked.cc/reference>. [Geopend 5 March 2024].
- [8] Nuxt, „Nuxt documentation,” Nuxt, [Online]. Available: <https://nuxt.com/docs/getting-started/installation>. [Geopend 10 06 2024].
- [9] ThreeJs, „ThreeJs.org,” [Online]. Available: <https://threejs.org/docs/index.html#manual/en/introduction/Installation>. [Geopend 25 4 2024].
- [10] ThreeJS, „ThreeJS Examples,” [Online]. Available: <https://threejs.org/examples/>. [Geopend 25 4 2024].
- [11] Minecolonies, „Minecolonies Wiki,” [Online]. Available: <https://minecolonies.com/wiki>. [Geopend 10 6 2024].
- [12] „[json.org](https://www.json.org),” json.org, 12 1999. [Online]. Available: <https://www.json.org/json-en.html>. [Geopend 25 03 2024].
- [13] SirEndii, „Advanced Peripherals Wiki,” [Online]. Available: <https://docs.advanced-peripherals.de>. [Geopend 19 March 2024].
- [14] Draxxusc3, „appliedenergistics.org,” appliedenergistics.org, 27 06 2014. [Online]. Available: <https://appliedenergistics.org/ae2-site-archive/ME-Controller/index.html>. [Geopend 2024].

Bijlagenoverzicht

Bijlage 1: Logboek rapporteren

Week	Naam student	Paginnummers	Taak
1	Thibe Provost		Verantwoordelijkheid gekregen voor verslagen tijdens de vergaderingen.
1	Pjotr Brunain		
1	Luca Vandenweghe		
1	Jonas Van Kerkhove		
2	Thibe Provost	0-6	Koppen verzinnen en kft rapport, mail versturen, begonnen layout logboek.
2	Pjotr Brunain	3	Inleiding maken
2	Luca Vandenweghe	3	Inleiding maken
2	Jonas Van Kerkhove	0-6	Koppen verzinnen en kft rapport
5	Thibe Provost	Volledig document	Fouten verbeterd a.d.h.v. feedback, inleiding herschreven
5	Pjotr Brunain	3 12-15	inleiding herschreven Eerste hoofdstuk: LUA en Minecraft
5	Luca Vandenweghe	3	inleiding herschreven Fouten gehaald uit eerste hoofdstuk
5	Jonas Van Kerkhove	3 12-15	inleiding herschreven Eerste hoofdstuk: LUA en Minecraft
6-9	Thibe Provost	Volledig document	Opkuisen document, verslagen toegevoegd in de bijlage, Hoofdstuk 4 geschreven
6-9	Pjotr Brunain	16-20	Proof-reading van Hoofdstuk 5
6-9	Luca Vandenweghe	16-20	Hoofdstuk 4 geschreven
6-9	Jonas Van Kerkhove	28-38	Hoofdstuk 5 geschreven
10-12	Thibe Provost	13-23	Hoofdstuk API geschreven en Backend ontwikkeling, logboek in juiste vorm gezet, code fragmenten verwijzing fouten opgelost
10-12	Pjotr Brunain	12-20	Deel hoofdstuk Minecraft en Mods herwerkt Hoofdstuk 3.3 geschreven Hoofdstuk 1: Situering aangemaakt.
10-12	Luca Vandenweghe	28-38	Hoofdstuk 4 herlezen Conclusie geschreven Gecheckt op taalfouten en deze verbeterd Meerde zinnen omgezet naar tegenwoordige tijd
10-12	Jonas Van Kerkhove	18-20	Hoofdstuk 3 geschreven
13	Thibe Provost	Volledig document	Herlezen document
13	Luca Vandenweghe	Volledig document	Herlezen document
13	Pjotr Brunain	Volledig document	Herlezen document
13	Jonas Van Kerkhove	Volledig document	Herlezen document

Bijlage 2: Verslag1

28/02/2024

Aanwezigen: Pjotr Brunain, Luca Vandeweghe, Jonas Van Kerkhove, Evert-Jan Jacobs, Thibe Provost.

Afwezigen: /

Wat hebben we allemaal al gedaan?

We hebben ter voorbereiding op deze vergaderingen een subgroup op gitlab.com gemaakt. Hierin hebben we alvast 3 projecten aangemaakt: één voor alle documentatie, één voor de website en één voor de applicatie. In ieder project maakten we ook meerdere issues aan zodat wij een goed overzicht konden krijgen van alle openstaande opdrachten. Het logboek en het rapport werden aangemaakt voor de vergadering en zal verder worden aangevuld naarmate de opdracht wordt uitgevoerd.

Wat kan beter?

Waar we zeker op moeten letten in de toekomst is het nalezen van documenten op schrijffouten en zinsstructuur. Het issue board op gitlab.com is ook nog steeds niet 100% volledig. Dit zouden we moeten proberen zo volledig mogelijk te maken zodat we een goed overzicht krijgen over de openstaande problemen. Ook de communicatie tussen de mentor en teamleden om een datum vast te leggen zou beter kunnen. Als groep zouden we beter een paar datums afspreken die wij kunnen en die voorleggen aan de mentor. De voorbereiding op de vergadering zelf kan vlotter verlopen. Voor de volgende keer kunnen we best al eens op voorhand samenzitten en nadenken wat er de volgende 2 weken moet gebeuren.

Wat moet er nog gebeuren?

- Een schema maken die visueel de samenwerking tussen de verschillende delen van ons project voorstelt.
- Het logboek moet eens nagelezen worden en alle fouten moeten eruit gehaald worden.
- Logboek aanvullen over week2.
- Verslag lay-out maken voor volgende vergadering.
- Kleine demo maken voor volgende vergadering.
- Issue board vervolledigen.
- Een extra label extra's maken om features bij te houden die tof zijn als uitbreiding. Bv: Prioriteiten aanpassen van de verschillende taken, kijken om het project in een server omgeving werkend te krijgen.
- Brainstormen over front-end.

Wat moet zeker gebeuren tijdens deze sprint?

- Het logboek aanvullen.
- Nadenken over een demo.
- Brainstormen over front-end.
- Nadenken over wat we de volgende 2 weken gaan doen.
- Uitnodiging naar mentor sturen voor volgende vergadering.

Tips van Evert-Jan

- Probeer het zo simpel mogelijk uit te leggen.
- Documenteer alles zodat iemand jullie project kan overnemen of als er een vervolgproject zou volgen jullie verder kunnen werken.
- Maak al een voorbereidend verslag zodat jullie weten wat jullie tijdens de meeting willen tonen.

Bijlage 3: Verslag2

13/03/2024

Aanwezig: Pjotr Brunain, Luca Vandeweghe, Jonas Van Kerkhove, Evert-Jan Jacobs, Thibe Provost.

Afwezig: /

Wat hebben we gedaan?

Thibe had een eerste ruw databank design gemaakt en ondertussen alweer wat verfijnd door Pjotr en Jonas. Ook heeft Jonas al een werkende demo gemaakt waarin we de data uit een voorbeeld json bestand halen. Pjotr, Jonas en Luca maakten een schema dat visueel de samenwerking tussen de verschillende delen van ons project voorstelt. Ook langs de Minecraft kant van ons project heeft Pjotr al wat werk verricht. Zo wordt er al data geëxporteerd aan de hand van een paar LUA-scripts. Ook voor de website heeft Luca al eens nagedacht over een design. Het issue bord werd uitgebreid met een “Need Feedback” kolom door Thibe en extra issues. Thibe heeft ook wat probleempjes in Gitlab opgelost.

Demo's

- Korte demo van het uitlezen van een Json bestand (Jonas)
- Korte demo LUA (pjotr)
- Kort voorstellen van schema

Wat kan beter?

- Demo's zijn goedgekeurd maar structureer het schema net iets beter
- Alle velden met kleine letters zetten in databank (Fk-velden)
- Probleem id dat hoog oploopt → Misschien een andere unieke key (misschien de id uit de Json file)
- Commentaar bij LUA
- Markdown bestandje maken op git

Wat doen we tegen volgende sprint?

- Get Items
- Db connectie
- Patterns aka recipies

- Extracten crafting recipies
- Final db shema
- Finish Figma design
- First step of website
 - Homepage en login

Tips van Evert-Jan

- Maak een kalender/ planning voor wie wat doet tot het einde van het project → via roadmap op git
- Wat gaan we tonen op het einde van de opdracht? Wat is ons eindproduct?

Bijlage 4: Verslag3

27/03/2024

Aanwezigen: Pjotr Brunain, Luca Vandeweghe, Jonas Van Kerkhove, Evert-Jan Jacobs, Thibe Provost.

Afwezigen: /

Wat hebben we gedaan?

Tijdens deze sprint hebben Jonas en Pjotr samen met een beetje hulp van Thibe de inleiding herschreven. Thibe heeft aan de hand van de gegeven feedback het rapport verbeterd. Pjotr heeft een hoofdstuk geschreven met veel feedback van Jonas. Jonas heeft ook nog verder aan de wpf-applicatie gewerkt, hij heeft de items van een request uit een Json file gehaald. Ook heeft hij dit gelinkt aan de correcte colonie en het in een mooie Gui gepresenteerd. Pjotr heeft geluisterd naar Evert-Jan en commentaar bij zijn LUA-code geschreven. Hij hielp ook Jonas met de wpf-applicatie door te beginnen aan de file selection . Hij heeft ook in Lua een script geschreven om de items uit te lezen en een epic en roadmap op git gemaakt. Thibe heeft dezelfde Json files omgezet in een Laravel project. Hij heeft migrations en seeders gemaakt om de databank te maken. Door gebrek aan tijd is het hem niet gelukt om de api routes te maken. Luca heeft de volledige website gemaakt en figma afgewerkt. Hij heeft zowel een mobile als desktopversie gemaakt en meerdere css animaties.

Demo's

Website → Luca

Wpf demo → Pjotr

Laravel → Thibe

Wat kan beter?

- 2 nav bars website
- Complexiteit website
- Automatisch genereren seeders?
- Mooi maken wpf?

Wat doen we tegen volgende sprint?

- Zie Roadmap Git

Tips van Evert-Jan

- Overzicht waar we willen eindigen
- Wanneer klaar?
- Dashboard
- Denk na over verhaal → maak filmpje/ demo/ iets
- Pjotr dacht om een aparte computer de demo te doen een eigenlijk live demo

Bijlage 5: Verslag4

17/04/2024

Aanwezigen: Pjotr Brunain, Luca Vandeweghe, Jonas Van Kerkhove, Evert-Jan Jacobs, Thibe Provost.

Afwezigen: /

Wat hebben we gedaan?

Tijdens deze sprint heeft Thibe gewerkt aan de api. Hij heeft al een post en get endpoint voor users, world, colonies en een get voor builderrequests met zijn location en specifiedrequest. Hij was ook begonnen met een delete maar dat bleek niet nodig. Door de sprintlabo's van web en andere verplichtingen heeft hij nog niet verder kunnen werken. Pjotr heeft commands in Lua werkend gekregen en de wrap peripherals script herwerkt hiervoor. Ook heeft hij een main loop voor het lua programma geïmplementeerd en een savestate voor lua geïmplementeerd. Jonas heeft Thibe erop gewezen dat de databank moest aangepast worden en hij heeft requests omgezet naar commands die aan lua worden overgedragen. Luca is nuxt beginnen leren. Zijn eerste indruk was: "wtf da is ". Ook heeft hij een nuxt project aangemaakt en is hij begonnen aan de transfer van onze website naar nuxt framework.

Demo's

Lua → Pjotr

Laravel api → Thibe

Wat kan beter?

- Verslag van wetenschappelijk rapporteren (toch meer in detail)
- Location <-> builderRequest een op een maken

Wat doen we tegen volgende sprint?

- Zie Roadmap Git
- Api af en al wat request op de website krijgen
- Het hapbaar maken van ons idee/project

Tips van Evert-Jan

- Eens samen met evert-jan samenzitten met verslag
 - Code fragmenten uitleggen
- Zorg voor iets werkend op de opendeurdag
- Maak het behapbaar

Bijlage 6: Verslag5

8/05/2024

Aanwezigen: Pjotr Brunain, Luca Vandeweghe, Jonas Van Kerkhove, Evert-Jan Jacobs.

Afwezigen: Thibe Provost

Wat hebben we gedaan?

Thibe en Jonas hebben ervoor gezorgd dat de requests, builder request, colonie en werelden in de databank worden opgeslagen door de data vanuit het C# Programma door te sturen naar de web API. Voor de storage items had Thibe geen tijd om de route te maken dus dat is voor volgende sprint. Luca heeft verder gewerkt aan de website en heeft de Nuxt hell even links laten liggen en leren werken met three.js in de plaats. Pjotr heeft nog wat verder gewerkt aan de LUA-scripts zo heeft hij voor Jonas en Thibe een fingerprint toegevoegd om iedere colonie gemakkelijk te identificeren. Ook heeft hij ervoor gezorgd dat we de recepten van de verschillende items kunnen extraheren uit de verschillende modpacks.

Demo's

API & C# link → Jonas

Three.js → Luca

Wat kan beter?

- GIT moet verbeterd worden en in orde gezet worden

Wat doen we tegen volgende sprint?

- Thibe maakt een route voor storage items en fixt de authenticatie.
- Pjotr extract recipes van de Minecraft jar en begint met recipe calculations. Hiernaast gaat hij het LUA-script fixen zodat commando's niet meerdere keren kunnen uitgevoerd worden.
- Luca gaat kijken met Bart voor Nuxt, en begint aan het dashboard.
- Jonas maakt de post requests voor items in storage. Hierna gaat hij samen met Thibe authenticatie op het C# programma voorzien. Ook gaat hij kijken

Tips van Evert-Jan

- Naar Luca, Vraag rond Nuxt ook zeker dingen aan Bart, Als we het nog niet vinden best inderdaad overschakelen naar een gewone website maken zonder Nuxt.
- Doorvragen aan mevr. Martens waarom we een bepaalde score gekregen hebben. Zijn er dingen waarmee we rekening moeten houden in de toekomst? Zeker ook nog eens vragen verder voor bachelor proef.
- Gebruik ChatGPT om teksten te verbeteren. Laat zeker weten aan ChatGPT dat het over een wetenschappelijk rapport gaat.

Bijlage 6: Verslag6

22/05/2024

Aanwezigen: Pjotr Brunain, Luca Vandeweghe, Jonas Van Kerkhove, Evert-Jan Jacobs, Thibe Provost.

Afwezigen:

Wat hebben we gedaan?

Pjotr heeft de minecraft jar te pakken gekregen om de recipes eruit te extracten, en heeft ook de git repo's opgekuist, Thibe en Luca hebben hier de frontend en API repo opgesplitst in 2 verschillende repo's om de projecten gemakkelijker te kunnen plaatsen op plesk. Luca heeft het project overgezet naar nuxt en hij heeft het three.js concept verder uitgewerkt en bijna afgewerkt en ook de login/dashboard en dus verbinding met de backend bijna afgewerkt, ook heeft hij samen met Pjotr een eerste versie van een handleiding zitten maken. Thibe heeft de laatste normale endpoint afgemaakt en al een volledige API eerste versie (zonder authenticatie en autorisatie) op plesk gezet (<https://minecraftapi.thibeprovost.ikdoeict.be/api/>). Thibe heeft ook eens aan Joris Maervoet gevraagd wat ze het beste voor authenticatie zouden gebruiken. Joris zijn antwoord was dat we best met JWT-Tokens gaan werken want zelf is hij ook geen fan van Sanctum. Jonas heeft de items in de databank gekregen en auto complete gefixed.

Demo's

Luca → website & three.js

Pjotr → git

Wat kan beter?

- Jonas en Thibe moeten hun issues updaten
- Kleurcontrast
- Foto's voor items

Wat doen we tegen het einde?

Thibe:

JWT-token implementeren en Luca helpen met de API aan te roepen

Git issues updaten

Proberen icons bij de items zetten

Luca:

Login fixen met Thibe

Three.js afwerken

Dashboard

Stijling

Jonas:

Git issues updaten

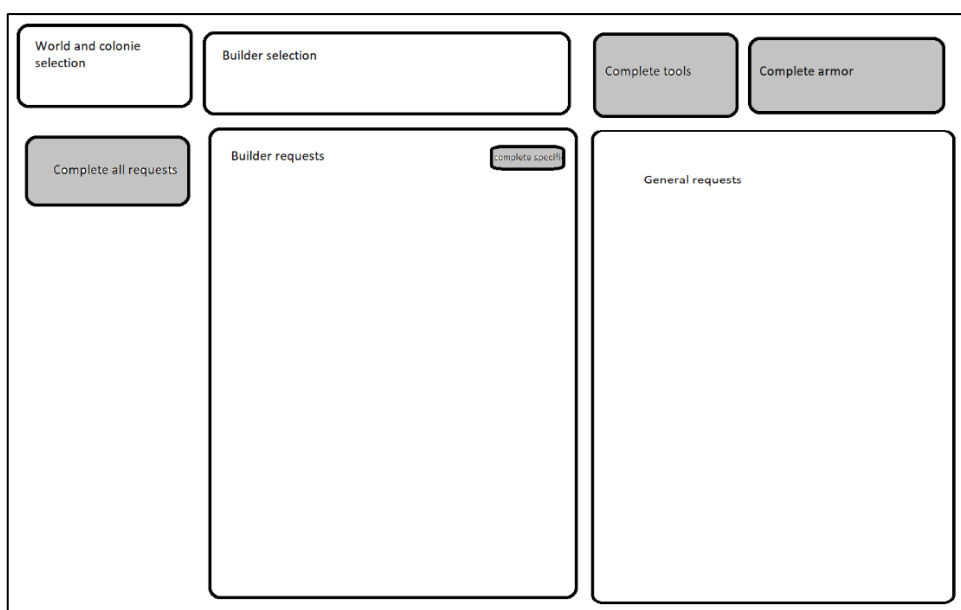
Post recipes

Pjotr:

Kijken om een installeren te maken en API op eigen server zetten

Tips van Evert-Jan

- Gaat er rol gebaseerde authenticatie inzitten? → neen aangezien dit niet nuttig is.
- Zet icons bij de items en denk na over uitbreiding
- Maak het dashboard net iets anders, teken het eens uit want voor iemand die het niet kent zegt dit niks. Maak eens een schets (Figuur 30)



Figuur 30: Schets dashboard

