FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Softwareproject

The Matrix

Team 10

Raman Talwar

Othello Clemens

Thomas Pellegrims

Gaétan Van Acker

Academiejaar 2020–2021

Bachelor of Science in de industriële wetenschappen: informatica

Valentin Vaerwyckweg 1, 9000 Gent

UNIVERSITEIT
GENT

Inhoud

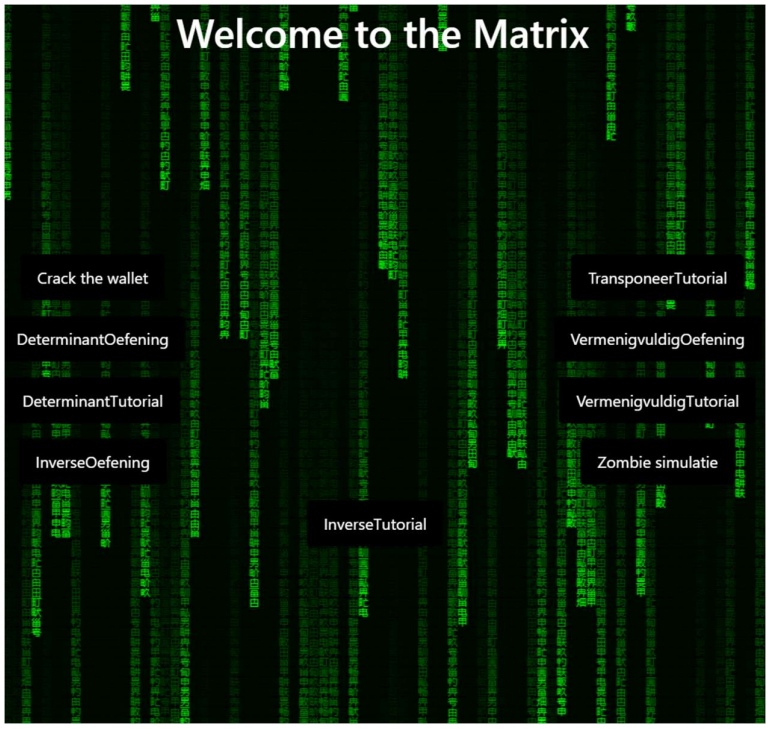
Inleiding 3
Use Case Diagrams 3
Algemeen 3
Use case 1: Matrixvermenigvuldiging aanleren 3
Use case 2: Interactieve matrixvermenigvuldigingen 4
Use case 3: Determinanten 4
Use case 4: Transponeren 5
Use case 5: Inverse 5
Use case 6: Toepassing 1: 'Decrypt the wallet' (Hill cipher encryptie) 6
Use case 7: Toepassing 2: Markov ketens 6
Sequentiediagrammen 8
MainPage 8
Tutorials 11
Oefeningen 13
Toepassing Bitcoins 15
Toepassing Zombie simulatie 17
Klassendiagram 18
Troubleshooting 20
Uitbreidingen en verbeteringen 21
Referentielijst 22

Inleiding

In deze analyse zal het project worden beschreven aan de hand van verschillende diagrammen. Dit project kan gebruikt worden door leerlingen om te leren werken met matrices en het nut ervan te ontdekken.

Het idee van dit project is om leerlingen kennis te laten maken met matrices en wat de verschillende bewerkingen ervan zijn. Dit wordt aangetoond met een stap-voor-stap uitleg die de gebruiker de bewerking aanleert. Daarna zijn er enkele oefeningen waarbij de leerling op een interactieve manier de matrixbewerkingen inoefent, en krijgt dan ook feedback. Er worden ook manieren voorzien om de gebruiker een hint te geven bij moeilijkheden.

Ten slotte zijn er enkele toepassingen waarmee gepoogd wordt de leerlingen de schoonheid van de aangeleerde theorie over matrices te laten inzien. Markovketens worden gebruikt om de evolutie van een bepaald systeem te modelleren. Daarna wordt er nog naar de moderne wereld gegrepen van de bitcoins om een wallet te decrypteren aan de hand van Hill cipher encryptie.



Figuur 1: MainPage

Use Case Diagrams

Algemeen

In het algemeen moet het project de theorie over verschillende basis matrix bewerkingen aanleren aan de gebruiker en moet de gebruiker verschillende oefeningen kunnen oplossen. Ook bevat het project 2 toepassingen op matrices om de gebruiker het nut van matrices te laten inzien.

Use case 1: Matrixvermenigvuldiging aanleren

- Naam:
    - Matrixvermenigvuldiging aanleren

- Doelstelling:
    - De gebruiker kan via een aantal stappen een matrix oplossen
- Actoren:
    - Gebruiker
- Precondities
    - Gebruiker moet begrip matrices verstaan
    - Gebruiker moet kunnen vermenigvuldigen en optellen
- Postcondities
    - Begrip matrices verstaan
    - Kennis matrixvermenigvuldiging en voorwaarden

- Successcenario
    - Gebruiker krijgt de algemene uitleg
    - Stap voor stap uitleg volgen
    - Gebruiker kan naar volgende stap door op knop te drukken
    - Na laatste stap verschijnt knop naar oefeningen
- Alternatief scenario
    - Gebruiker klikt op exit knop en komt op de mainpage terecht

Use case 2: Interactieve matrixvermenigvuldigingen

- Naam:
    - Interactieve matrixvermenigvuldigingen
- Doelstelling:
    - Gebruiker kan een aantal matrixvermenigvuldigingen uitvoeren
- Actoren:
    - Gebruiker
- Precondities
    - Tutorial matrixvermenigvuldiging uitgevoerd
- Postcondities
    - Matrices kunnen vermenigvuldigen
- Successcenario
    - Gebruiker krijgt 2 matrices
    - Gebruiker vult element per element het resultaat in
    - Na correct oplossen verschijnt knop naar volgende oefening
- Alternatief scenario
    - Gebruiker klikt op hint knop voor een hint
    - Gebruiker klikt op exit knop en komt op de hoofdpagina terecht

Use case 3: Determinanten

- Naam:
    - Determinant tutorial volgen en oefeningen oplossen
- Doelstelling:
    - Gebruiker heeft kennis over determinanten en kan ze uitrekenen
- Actoren:
    - Gebruiker

- Precondities
    - Basiskennis matrices
- Postcondities
    - Gebruiker heeft kennis over determinanten en kan ze uitrekenen
- Successcenario
    - De gebruiker krijgt algemene uitleg over determinant
    - Stap per stap bewerkingen met uitleg bekijken
    - Gebruiker leest de extra uitleg stap per stap
    - Gebruiker klikt op next knop om naar volgende stap te gaan
    - Gebruiker klikt op voltooien
    - Gebruiker krijgt een matrix
    - Gebruiker kiest juiste oplossing uit de gegeven oplossingen
    - Gebruiker krijg feedback
    - Gebruiker krijgt 2 analoge oefeningen
    - Felicitaties aan gebruiker na oplossen alle oefeningen
- Alternatief scenario
    - Gebruiker klikt op hint knop voor een hint
    - Gebruiker klikt op exit knop en komt op de hoofdpagina terecht

Use case 4: Transponeren

- Naam:
    - Uitleg matrix transponeren
- Doelstelling:
    - Gebruiker weet hoe hij matrices moet transponeren
- Actoren:
    - Gebruiker
- Precondities
    - Basiskennis matrices
- Postcondities
    - Gebruiker weet hoe hij matrices moet transponeren
- Successcenario
    - Algemene definitie over transponeren van matrices
    - Stap per stap animatie tonen
    - Gebruiker bekijkt de animatie en de extra uitleg stap per stap
    - Gebruiker klikt op next knop om naar volgende stap te gaan
- Alternatief scenario
    - Gebruiker klikt op exit knop en komt op de hoofdpagina terecht

Use case 5: Inverse

- Naam:
    - Inverse uitleg
- Doelstelling:
    - Gebruiker weet hoe hij matrices moet inverteren
- Actoren:
    - Gebruiker
- Precondities

    - Determinanten kunnen uitrekenen en matrices kunnen transponeren
- Postcondities
    - Gebruiker weet hoe hij matrices moet inverteren
- Successcenario
    - Algemene definitie over inverteren van matrices
    - Stap per stap bewerkingen uitleggen (adjunct berekenen)
    - Gebruiker klikt op next knop om naar volgende stap te gaan
    - Gebruiker klikt op voltooien
    - Gebruiker krijgt matrix
    - Gebruiker stelt zelf de inverse op
    - Gebruiker krijgt feedback
    - Gebruiker krijgt 2 analoge oefeningen
    - Felicitaties aan gebruiker na oplossen alle oefeningen
- Alternatief scenario
    - Gebruiker klikt op hint knop voor een hint
    - Gebruiker klikt op exit knop en komt op de hoofdpagina terecht

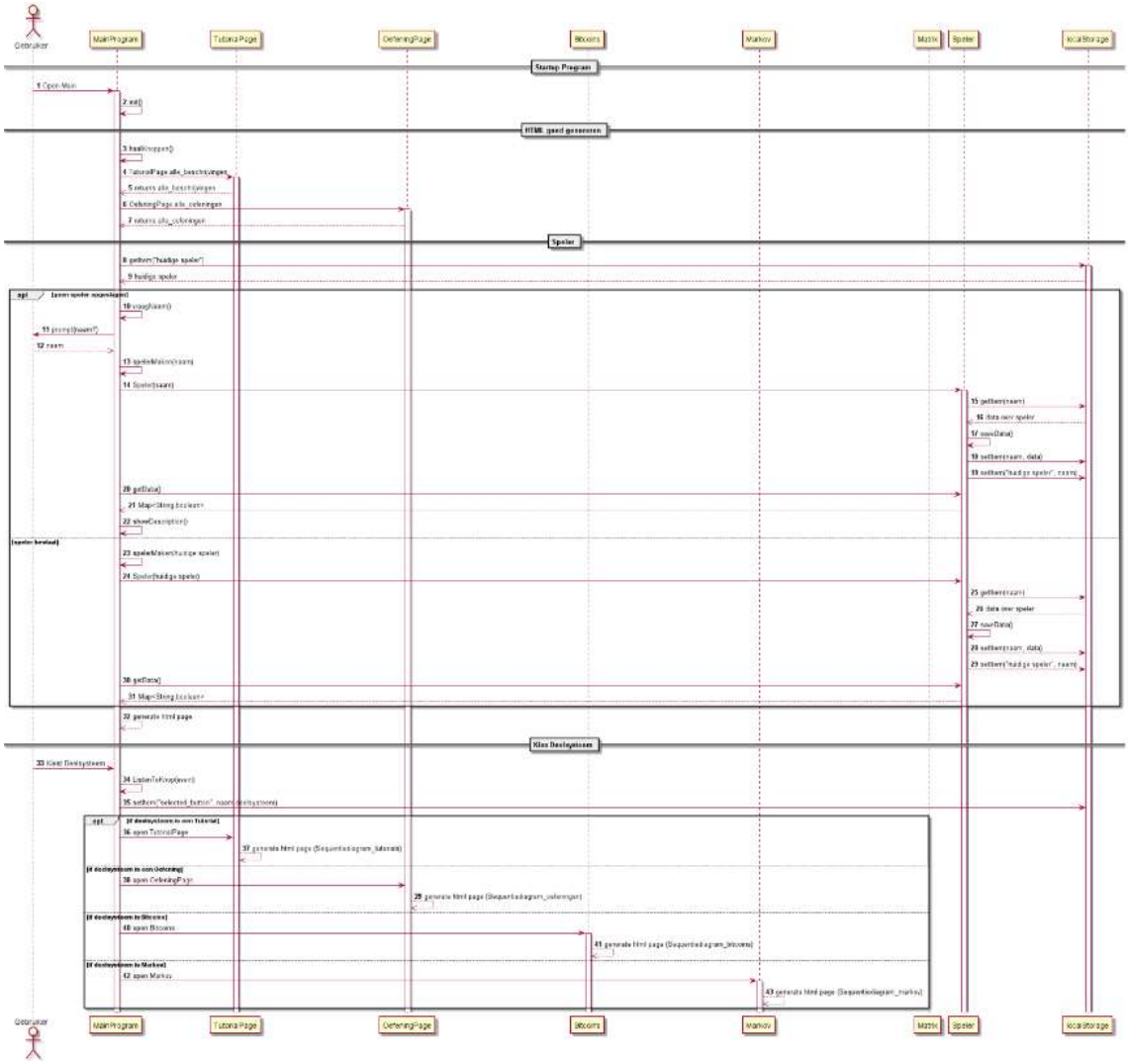
Use case 6: Toepassing 1: 'Decrypt the wallet' (Hill cipher encryptie)

- Naam:
    - Toepassing: Decrypt the wallet
- Doelstelling:
    - Gebruiker kan opgestelde wachtwoord van de wallet ontcijferen
- Actoren:
    - Gebruiker
- Precondities
    - Gebruiker heeft alle tutorials uitgevoerd en begrepen
- Postcondities
    - Gebruiker kan letters aan de hand van matrix encrypteren en decrypteren (Hill cipher encryptie toepassen)
- Successcenario
    - Gebruiker krijgt verhaal over bitcoinwallet
    - Gebruiker krijgt uitleg over Hill cipher encryptie
    - Gebruiker klikt op start
    - Gebruiker krijgt versleutelingsmatrix en gecodeerde key
    - Timer start
    - Gebruiker berekend de inverse van de versleutelingsmatrix
    - Gebruiker ontcijfert de key
    - Gebruiker geeft de juiste key in
    - Timer stopt
- Alternatief scenario
    - Timer loopt af
        - Er wordt een nieuwe key gegenereerd die de gebruiker moet ontcijferen
    - Gebruiker klikt op hint knop om hints te bekijken

Use case 7: Toepassing 2: Markov ketens

- Naam:
    - Toepassing: in markov ketens

- Doelstelling:
    - Gebruiker begrijpt het belang van matrices in markov ketens
- Actoren:
    - Gebruiker
- Precondities
    - Gebruiker heeft alle tutorials uitgevoerd en begrepen
- Postcondities
    - Gebruiker begrijpt het belang van matrices in markov ketens
- Successcenario
    - Uitleg over markov ketens
    - Gebruiker krijgt een verhaal om de simulatie te kaderen
    - Gebruiker kan interactief evolutie van systeem bekijken
    - Gebruiker kan zelf de overgangsmatrix aanpassen om het effect te bekijken
- Alternatief scenario
    - Gebruiker klikt op exit knop en komt op de hoofdpagina terecht



Figuur 2: Sequentiediagram mainpage

Sequentiediagrammen

MainPage

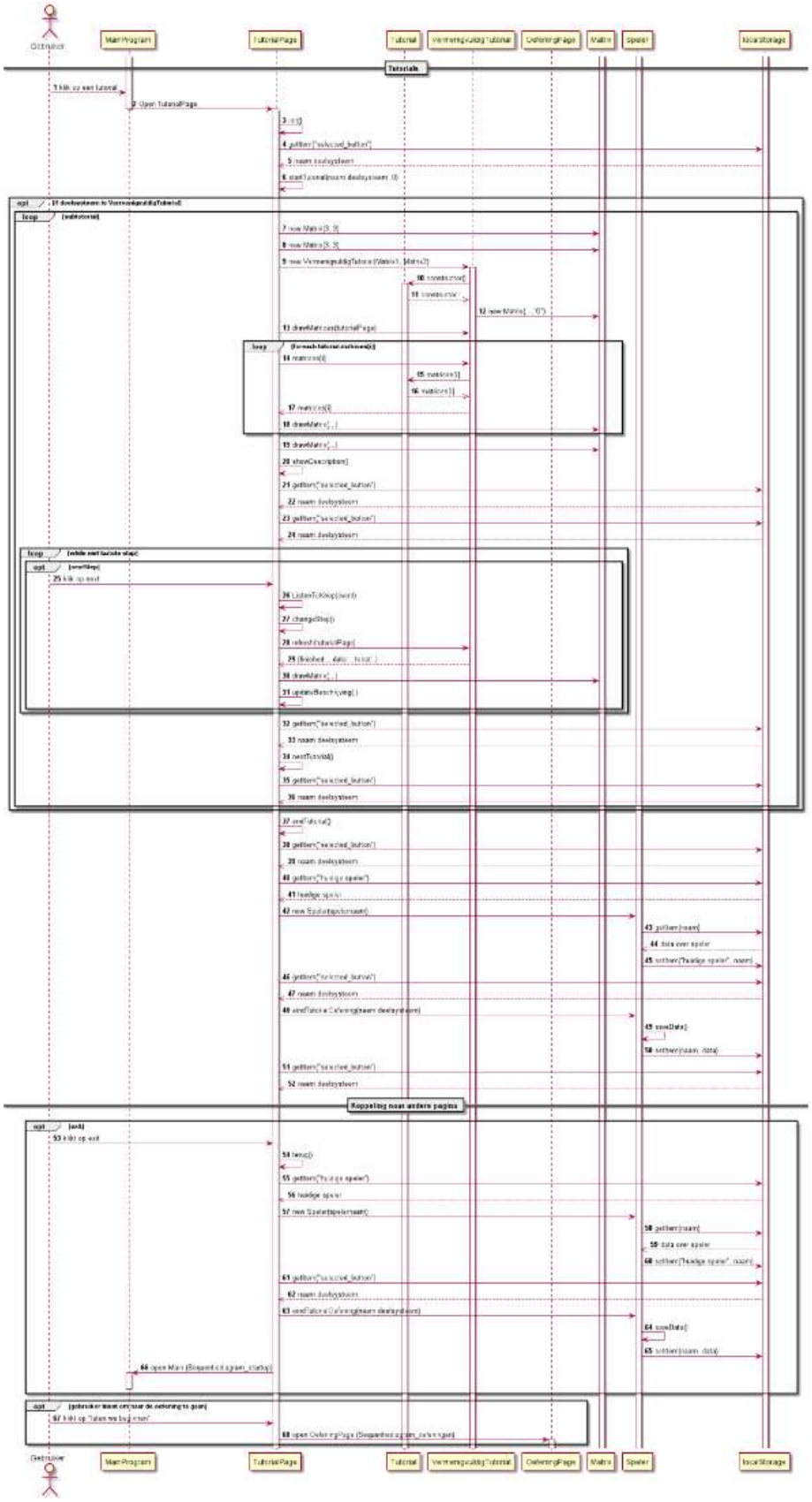
Het project begint bij de MainPage. Bij het laden van de pagina wordt er een script geladen die zorgt voor een dynamische achtergrond. Ook worden de knoppen voor de tutorials en oefeningen opgehaald en weergegeven. Deze knoppen krijgen allen een naam, welke worden opgehaald uit de klasse TutorialPage en Oefeningenpage. Hierbij worden twee extra knoppen aangemaakt die beide wijzen naar een toepassing namelijk: “Zombie simulatie” die wijst naar een toepassing van Markov ketens en “Crack the wallet” die wijst naar de Hill Cypher toepassing.

Wanneer de gebruiker voor het eerst de site bezoekt wordt een gevraagd naar de naam. Aan de hand van LocalStorage zal een map van gebruikers bijgehouden worden, Van elke gebruiker wordt bijgehouden welke onderdelen hij al heeft voltooid.

Indien de gebruiker eerder heeft deelgenomen aan het programma, worden de voordien aangemaakte gegevens opgehaald.

Nu kan de gebruiker een deelsysteem kiezen waarmee er wordt gestart, door op een van de gegenereerde buttons te klikken. De data van de aangeklikte button wordt meegegeven aan de LocalStorage vooraleer de speler verder wordt gestuurd. Er zijn twee soorten deelsystemen waar de gebruiker aan kan deelnemen.

Wanneer er op een button geklikt wordt van het type “Tutorial”, zal de HTML van TutorialPage zich genereren. Als dit van het type “Oefening” is, zal de HTML van de OefeningPage worden gegenereerd. De gebruiker kan evenals kiezen voor één van de twee toepassingen.



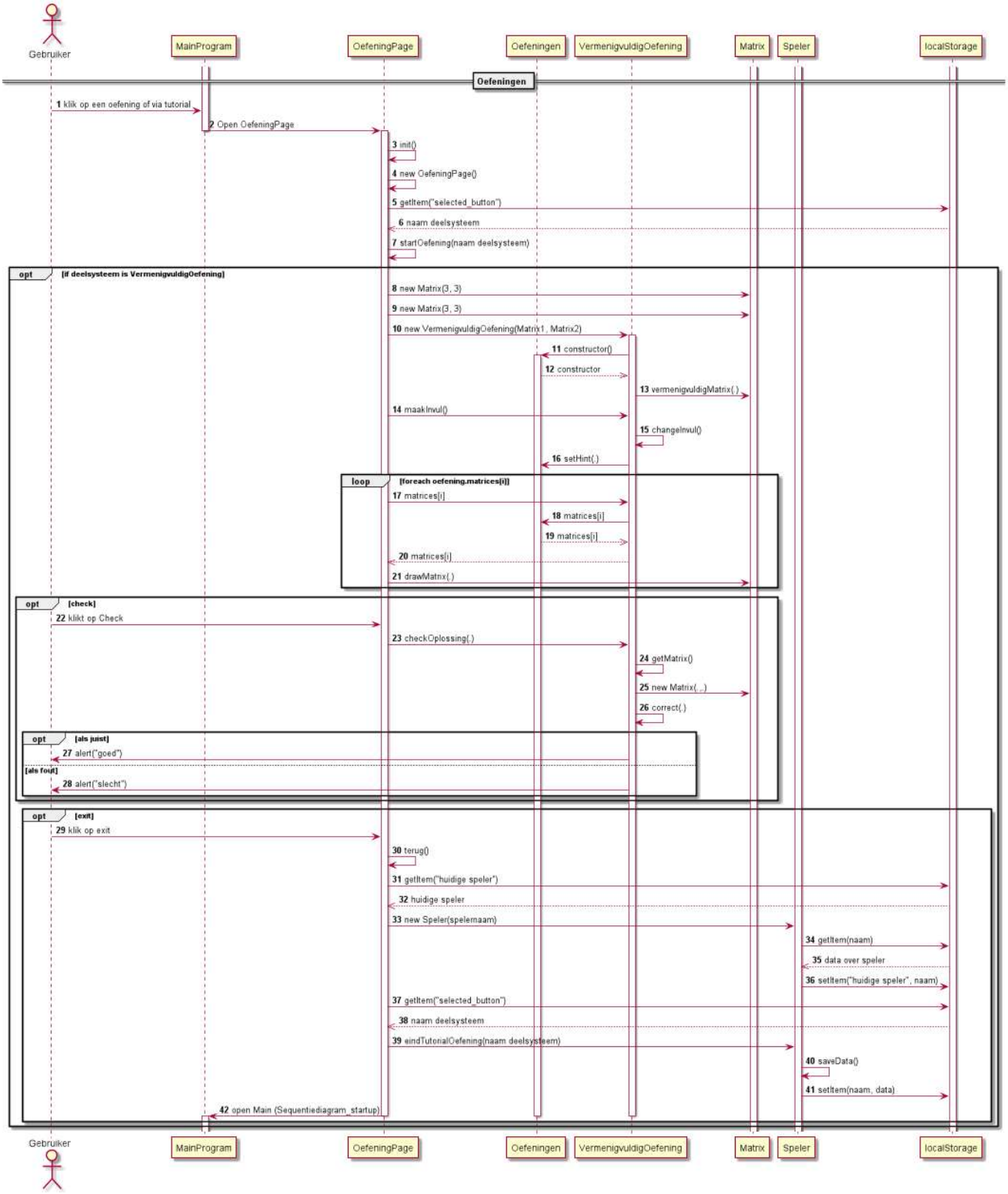
Figuur 3: Sequentiediagram Tutorials

Tutorials

Wanneer de gebruiker de keuze heeft gemaakt om een Tutorial te starten, zal de TutorialPage de data van de geselecteerde button opvragen aan de LocalStorage waaruit deze de juiste Tutorial kan starten. In het sequentiediagram staat één van de Tutorials uitgewerkt, namelijk de VermenigvuldigTutorial. In de TutorialPage worden twee objecten aangemaakt van het type Matrix. Deze worden meegegeven aan de contructor van een object van de klasse VermenigvuldigTutorial. Hierin wordt een nieuwe matrix aangemaakt waarin de uitkomst van de vermenigvuldiging gevisualiseerd zal worden. De verschillende matrices die gebruikt worden door de tutorial worden in een array opgeslagen. Met deze array kan de TutorialPage, de matrices op de juiste plaats in het HTML bestand plaatsen. Na het tekenen van de matrices wordt de uitleg van het gekozen deelsysteem ook gegenereerd.

Nu kan de gebruiker kiezen tussen de exit en de next button. Zolang dat de Tutorial niet aan het einde is gekomen kan de gebruiker de next button gebruiken om de volgende stap van het deelsysteem opvragen. Hierbij worden beide de matrix en de beschrijving van de huidige stap veranderd naar deze van de volgende stap. De gebruiker kan op elk ogenblik van de Tutorial de exit knop gebruiken. Deze knop zal de data van de speler bijwerken en vervolgens de speler terug leiden naar de MainPage.

Wanneer echter de speler aan het einde van de Tutorial komt zal een volgende Tutorial starten met nieuw willekeurig gegenereerde inhoud. Op het einde van de reeks Tutorials past de data van de speler aan waarin meegegeven wordt dat het bekeken deelsysteem volledig is afgerond. Vervolgens krijgt de speler een melding die meegeeft dat alle Tutorials van dit deelsysteem zijn afgewerkt en hij verder kan gaan naar de oefeningen die betrekking hebben op deze Tutorials. De speler kan dit weigeren door de prompt te sluiten. Ten slotte is er enkel de mogelijkheid om de exit button aan te klikken, en zo te worden herleid naar de MainPage.



Figuur 4:Sequentiediagram Oefeningen

Oefeningen

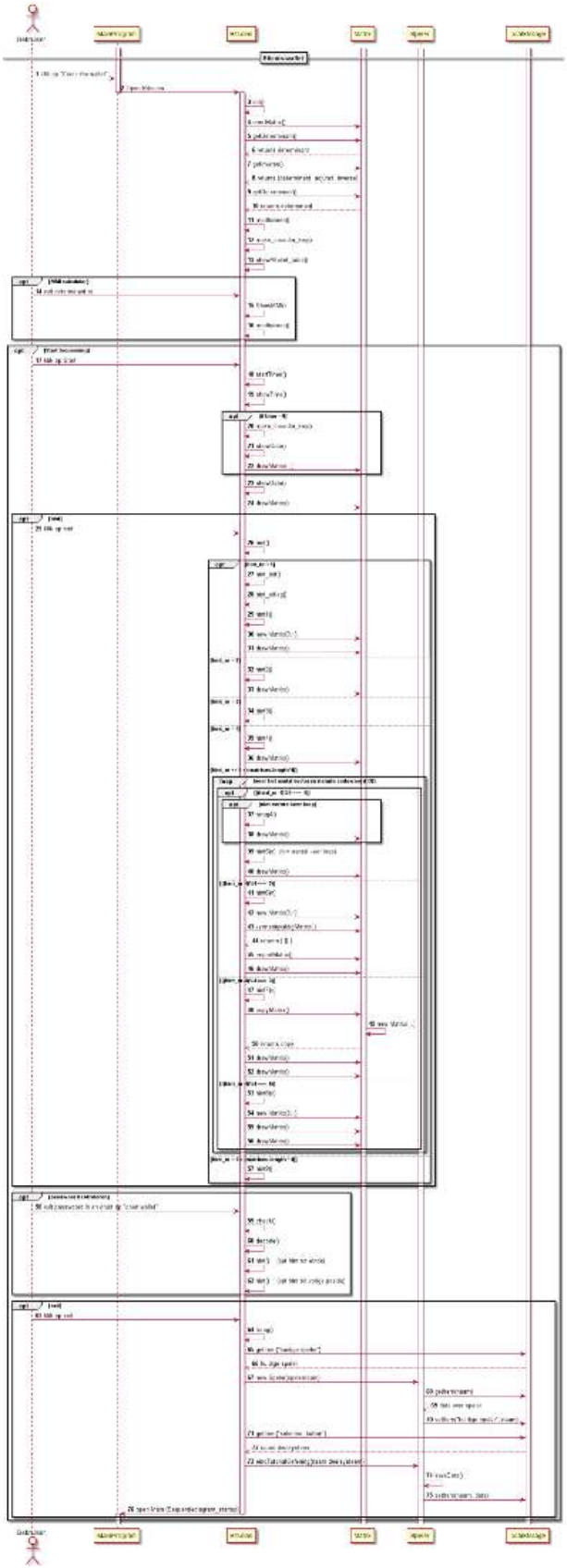
Wanneer de speler op de MainPage kiest om een deelsysteem van het type “Oefening” te starten, zal de OefeningenPage geopend worden. Dit kan ook gebeurden wanneer de Speler de corresponderende tutorial heeft afgewerkt.

Na het aanmaken van een nieuwe OefeningPage, zal de data van het eerder gekozen deelsysteem opgevraagd worden aan de LocalStorage. Aan de hand van deze data zal het juiste type oefening gestart worden.

In het sequentiediagram staat het voorbeeld voor het type “VermenigvuldigOefening”. In de OefeningPage worden een aantal willekeurig gegenereerde matrices aangemaakt en meegegeven aan de constructor van VermenigvuldigOefening. De VermenigvuldigOefening zal een nieuwe matrix aanmaken en laten verwijzen naar de uitkomst van de oefening die in de klasse Matrix wordt uitgevoerd. De meegegeven matrices van de OefeningPage zullen uitgetekend worden op de HTML. Bij VermenigvuldigOefening is gekozen voor een invuloefening, waarbij de speler de grootte van de matrix kan aanpassen en zijn gevonden uitkomst kan invullen.

Er worden 3 knoppen voorzien, een hint knop een check knop en een exit knop. De check button zal een “checkOplossing” methode oproepen die de ingevulde matrix zal vergelijken met de eerder gegenereerde uitkomst. Wanneer deze vergelijking niet klopt, zal er een tekst verschijnen op het scherm. Bij een juist antwoord wordt er een prompt getoond en start de volgende oefening.

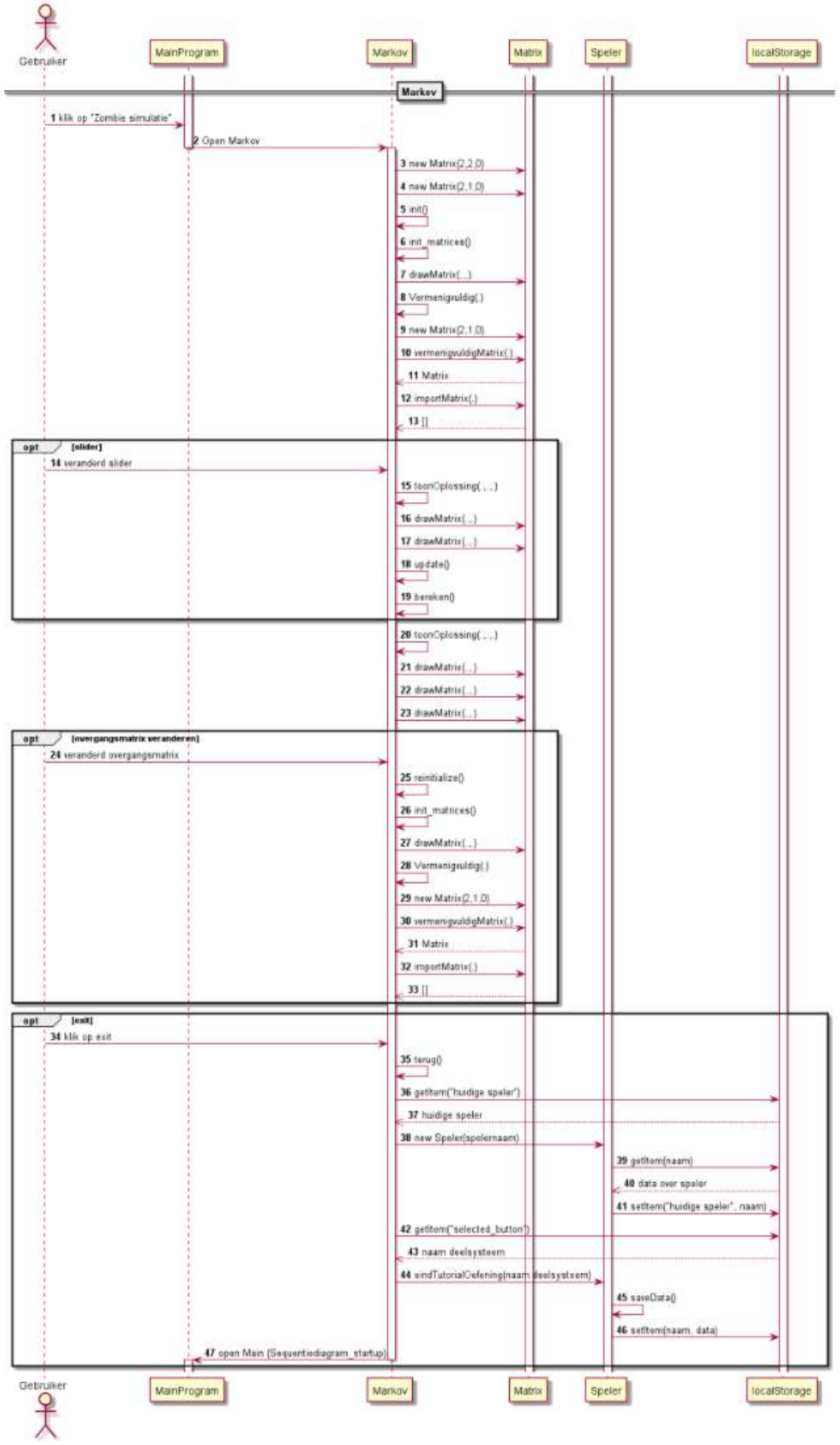
Na het oplossen van alle oefeningen in de reeks wordt de data van de speler aangepast in de LocalStorage. Ten slotte wordt de speler teruggestuurd naar de MainPage waar opnieuw een deelsysteem geselecteerd kan worden.



Figuur 5: Sequentiediagram Bitcoins

Toepassing Bitcoins

Ten slotte worden de twee toepassingen besproken beginnend bij de “Bitcoin crack the wallet” toepassing. Wanneer de speler op de gerelateerde button klikt, wordt de BitcoinPage geopend. Hier zal een willekeurige matrix aangemaakt worden. Daarbij zullen ook de verschillende methodes op deze matrix opgeroepen worden. Al deze methodes hebben een return waarde die evenals bijgehouden worden in de klasse. Na het genereren van deze attributen, zal de alfabet tabel uitgetekend worden. Vervolgens kan de speler op de startknop klikken om de toepassing effectief te starten. Wanneer op de startknop wordt geklikt, start een timer die weergeeft hoeveel tijd de speler over heeft om de code te ontcijferen. Verder wordt de rest van de HTML gegenereerd waarop de matrix wordt getekend. Nu moet de speler proberen de code te ontcijferen en kan hiervan gebruik maken van één van de negen toegevoegde hints. Wanneer de speler op de “open wallet” button klikt wordt zijn oplossing gecontroleerd. Ten slotte wordt een alert weergegeven waar de speler te weten kan komen of zijn oplossing juist was.

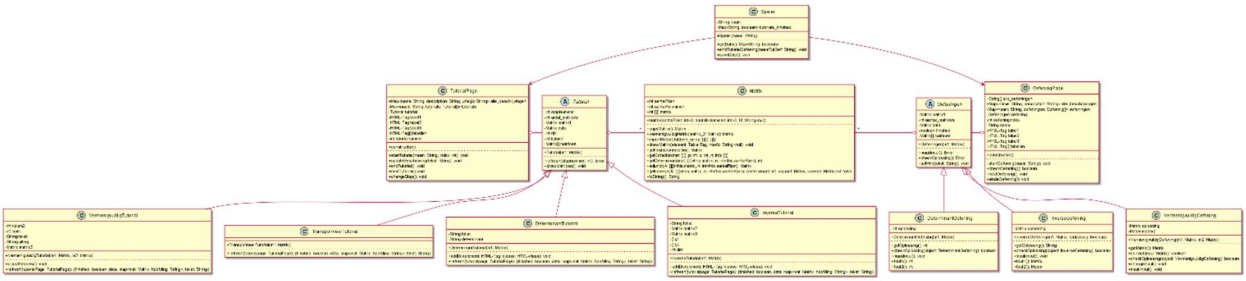


Figuur 6: Sequenteidiagram Zombie simulatie

Toepassing Zombie simulatie

De tweede toepassing, de “Zombie simulatie”, is een uitwerking van Markov ketens. Wanneer de speler de “Zombie simulatie” knop aanklikt op de MainPage, wordt de Markov pagina geopend. Er worden beide een overgangsmatrix en toestandsmatrix aangemaakt die worden weergegeven op de HTML page. Vervolgens wordt een matrix aangemaakt waarin de oplossing van de Markov keten zal in worden gevisualiseerd. De speler kan twee parameters veranderen op het scherm. Enerzijds kan de slider versteld worden. De slider geeft het aantal keer dat de vermenigvuldiging van de matrices heeft plaatsgevonden aan. Wanneer de slider versteld wordt gebeurt de vermenigvuldiging van de ingevulde overgangsmatrix met de toestandsmatrix die is weergegeven op het scherm. Hierbij worden de overgangsmatrix en de oplossing herberekend en, samen met de gevisualiseerde grafiek, opnieuw uitgetekend op de HTML page. Tot slot kan de gebruiker kiezen om de getallen in de overgangsmatrix aan te passen. Bij elke verandering wordt de toestandsmatrix wederom geïnitialiseerd. Daarop volgt dat de oplossing van de voorgaande matrices nogmaals aangemaakt en ingevuld wordt. Ter afsluiting kan de speler eveneens op de zichtbare exit button klikken, die hetzelfde realiseert als de exit knop bij de “Crack the wallet” toepassing.

Klassendiagram



Figuur 7: Klassendiagram

De klasse Speler houdt de naam van de huidige gebruiker en de oefeningen/tutorials die hij afgewerkt heeft. De klasse heeft een constructor waar de ingegeven naam van de huidige gebruiker meegegeven wordt, die gaat dan kijken of de speler al bestaat of niet en zo nodig een nieuwe aanmaken. De methode saveData() gaat de data van de huidige gebruiker doorgeven aan de LocalStorage. De methode eindTutorialOefening(naamTutOef) wordt opgeroepen als een tutorial/oefening/toepassig gedaan is en slaat die informatie via saveData() in de LocalStorage op. Er is ook een getter voor de data.

De klasse TutorialPage houdt alle beschrijvingen van alle tutorials bij in de Map alle_beschrijvingen en alle tutorial objecten worden bijgehouden in de Map tutorials. De variabele tutorial houdt de huidige tutorial bij en tutorialnummer houdt bij welk deel van de tutorial open staat. De variabelen tabel1-3 worden ook bijgehouden in de array tabellen en worden gebruikt om gemakkelijk de inhoud van de tabellen te wijzigen.

De klasse heeft een methode startTutorial die als parameters de naam van de specifieke tutorial en de index van het deel meekrijgt, de methode genereert de basis html. De klasse heeft ook een methode updateBeschrijving die de beschrijving zal updaten. De methode endTutorial zal de juiste modal oproepen en de spelerklasse informeren dat de tutorial klaar is. nextTutorial zal de modal oproepen om naar het volgende deel te gaan. changeStep is de methode die zorgt dat de juiste methodes worden opgeroepen om naar de volgende stap te gaan, als dat kan.

De abstracte klasse Tutorial heeft een attribuut stapnummer waarin hij het stapnummer bijhoudt. matrix1 is de beginmatrix van alle tutorials, ze wordt verkregen via de constructor. Het attribuut data is van de vorm {finished: boolean, data: matrix, tekst: "beschrijving"}, data wordt teruggegeven door de overschreven refresh methodes. De attributen rij1 en kolom1 worden gebruikt om gemakkelijk de huidige rij en kolom aan te spreken van matrix1. De array matrices wordt gebruikt om alle gebruikte matrices samen bij te houden.

De klasse heeft een constructor waar een matrix aan gegeven wordt die als basis dient voor de tutorial. De klasse heeft ook een methode refresh(stapnummer) waar het stapnummer aan meegegeven wordt en die overshreven moet worden bij elke tutorial omdat de stappen natuurlijk anders zijn. De methode drawMatrices() maakt een deel van de html klaar om ingevult te worden.

De klassen VermenigvuldigTutorial, TransponeerTutorial, InverseTutorial en DeterminantTutorial zijn afgeleide klassen van Tutorial. Deze klassen overschrijven allemaal de methode refresh omdat voor elke tutorial de stappen anders zijn. Bij vermenigvuldigTutorial is drawMatrices() overschreven omdat daar de volgorde anders is. DeterminantTutorial en InverseTutorial hebben ook nog de methode addDiv(element,classe) die een div met klasse classe aanmaakt en het bij element toevoegt.

Ze hebben elk hun eigen attributen die nodig zijn voor het goed functioneeren van de klasse. De klassen hebben ook nog hun constructoren die, met de hulp van de constructor van Tutorial, de nodige matrices aanmaaken en de attributen initialiseren.

De klasse OefeningPage is gelijkaardig aan de klasse TutorialPage maar de methoden die met stappen te maken hebben worden vervangen door de methode checkOefening() die teruggeeft of de oefening juist is.

De abstracte klasse Oefeningen is gelijkaardig aan de klasse Tutuorial, de methoden zijn wel helemaal anders. De methode maakInvul() moet overschreven worden maar zal de inputmethode aanmaken om de oefening te kunnen antwoorden. De klasse heeft ook een methode showHint(tekst) die de hint zal tonen. En een methode checkOplossing die overschreven moet worden maar zal de ingegeven antwoorden controleren en met een boolean teruggeven of het juist is.

De klassen DeterminantOefening, InverseOefening en VermenigvuldigOefening zullen elk de methode checkOplossing overschrijven doordat telkens andere methodes moeten opgeroepen worden van de klasse Matrix om de oplossing te vinden. Ze moeten elk ook de methode maakInvul() overschrijven omdat niet bij elke oefening dezelfde inputmethode gebruikt wordt.

Ze hebben ook nog elk hun constructor die de nodige matrices meekrijgt en bij InverseOefening is er ook de optie voor strikvragen aan of uit te zetten.

De klassen DeterminantOefening en InverseOefening hebben elk ook een methode getOplossing om gemakkelijk de oplossing te kunnen vergelijken met de input van de gebruiker. Ze hebben ook de methoden fout1 en fout2, deze methoden zijn er om op een dynamische manier veelgemaakte fouten tussen de antwoordmogelijkheden te steken.

De klasse VermenigvuldigOefening heeft een methode getMatrix() die een Matrix object maakt van de input van de gebruiker. Een methode correct die controleert of de input juist is. En een methode changeInvul die de inputvelden verandert als de gebruiker dat wilt.

De klasse Matrix houdt alles bij van een bepaalde matrix, dus het aantal rijen en kolommen en de matrix zelf in een dubbele array vorm.

De klasse heeft ook een constructor om een random matrix aan te maken, deze krijgt het aantal rijen en kolommen mee, de matrix heeft ook een optie om hem in te vullen met een bepaald karakter.

De klasse heeft een methode copyMatrix om een kopie van een matrix te krijgen. De methode importMatrix zal voor de gegeven dubbele array een matrixobject aanmaken. De methode

drawMatrix zal de matrix in de meegegeven table-tag steken. De methode toString zal de matrix in String vorm teruggeven om hem gemakkelijk in een uitleg of hint te tonen.

Daarnaast heeft de klasse ook nog een paar andere methodes die verschillende bewerkingen gaan uitvoeren om gemakkelijk de antwoorden op de oefeningen te vinden.

Troubleshooting

Voor Javascript bestaan er enkele 'testing frameworks' zoals Jest, Mocha, … Voor dit project werden er geen frameworks gebruikt voor het testen. De eerste manier waarop er code werd uitgetest, was om op verschillende plaatsten bepaalde waarden te laten uitschrijven via de console. Aan de hand van deze waarden kan er gecontroleerd worden of de code correct verloopt. Deze methode is vooral handig als men al een vermoeden heeft waar de bug zich ongeveer voordoet.

Een tweede manier was het gebruiken van de debugger via de IntelliJ IDE genaamd "JavaScript Debug". Hiermee kunnen er op verschillende plaatsten in de code 'breakpoints' gezet worden. Als dit stukje code wordt uitgevoerd, zal het programma pauzeren en geeft IntelliJ alle waarden die op dit ogenblik gebruikt worden. Daarna kan het verdere verloop van het programma stap voor stap gevolgd worden en kan de bug snel opgespoord worden.

De meest effectieve methode was de debugger, omdat deze zeer veel informatie geeft tegenover de eerste methode. Een nadeel van de debugger is dat deze niet zo snel resultaten oplevert tegenover de methode via de console.

Het gebruik van 'testing frameworks' zou dit proces vergemakkelijkt hebben. Maar het zou ook geen goed idee zijn om halverwege het project op een andere methode over te stappen. Om die reden zijn steeds de originele zoekstrategieën toegepast.

Uitbreidingen en verbeteringen

Het project is dynamisch ontworpen. Dit wil zeggen dat als er een nieuwe toepassing wordt toegevoegd, zal er weinig of geen code moeten aangepast worden in de al bestaande code. Dit zal het toevoegen van de eventuele uitbreidingen vergemakkelijken.

Er werd voorgesteld om via matrices het orthogonaliseren van kegelsneden aan te leren. Maar omdat dit niet aansloot bij het gekozen doelpubliek is dit niet uitgewerkt. Een andere uitbreiding is het aanleren van eigenwaarden en eigenvectoren van matrices. Deze uitbreiding is realistischer en kon uitgewerkt worden, maar er werd beslist om meer focus te leggen op de toepassingen van matrices. De reden hiervoor is om de gebruiker het 'nut' van matrices meer in te laten zien aan de gebruiker.

In de ontwerpfase werd er voorzien om een eigen klasse 'matrix.js' te maken in plaats van een al bestaande bibliotheek te gebruiken. Dit had natuurlijk het voordeel dat de code zelf snel kan worden aangepast, maar deze code is niets nieuws omdat er al bibliotheken met veel meer methodes al bestaan. Het gebruik van deze bibliotheek had het project kunnen versnellen omdat er dan niet gefocust moest worden op het zelf schrijven van deze bibliotheek.

Er zijn nog enkele verbeteringen mogelijk zoals het structureren van de toepassingen in verschillende klassen, maar er werd gekozen om de nadruk te leggen op de gebruikerservaring.

Referentielijst

Adjoint and Inverse of a Matrix. (2016, januari 20). *GeeksforGeeks*.

https://www.geeksforgeeks.org/adjoint-inverse-matrix/

Algorithmen verstehen. (2020, februari 18). *Modular Inverse of a Matrix | #Cryptology*.

https://www.youtube.com/watch?v=Rd2-EmS26uw&ab_channel=Algorithmenverstehen

Check if a Matrix is Invertible. (2018, augustus 22). *GeeksforGeeks*. https://www.geeksforgeeks.org/check-if-a-matrix-is-invertible/

Determinant of a Matrix. (2016, januari 8). *GeeksforGeeks*. https://www.geeksforgeeks.org/determinant-of-a-matrix/

Getransponeerde matrix. (2020). In *Wikipedia*.

https://nl.wikipedia.org/w/index.php?title=Getransponeerde_matrix&oldid=57040480

Hill cipher. (2021). In *Wikipedia*.

https://en.wikipedia.org/w/index.php?title=Hill_cipher&oldid=1011171191

How to Multiply Matrices. (z.d.). Geraadpleegd 8 april 2021, van

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Modular inverses (article) | Cryptography. (z.d.). Khan Academy. Geraadpleegd 16 april 2021, van

https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-inverses

Modular multiplicative inverse. (2015, juni 20). *GeeksforGeeks*.

https://www.geeksforgeeks.org/multiplicative-inverse-under-modulo-m/

Online calculator: Modular inverse of a matrix. (z.d.). Geraadpleegd 16 april 2021, van

https://planetcalc.com/3324/

Zach Star. (2019). *The Applications of Matrices | What I wish my teachers told me way earlier*. https://www.youtube.com/watch?v=rowWM-MijXU&t=822s&ab_channel=ZachStar

Matrix rain animation using HTML5 canvas and javascript. (z.d.). CodePen. Geraadpleegd 23 maart 2021, van https://codepen.io/P3R0/details/MwgoKv