**JMP Designdocument**

A black background with white dots

Description automatically generated

**Gemaakt door:**

**Mitchel Meskes**

**Joshua de Bruijn**

**Pedro Eduardo Cardoso**

**Datum: 11 Februari 2024**

**Inhoudsopgave**

- Inhoudsopgave

- Inleiding 3

- Versie beheer 4

- Belangrijke rollen 5

- Plan van Eisen 6

- Huisstijl 7

- User story’s 8, 9, 10, 11, 12

- Wireframe

- Mockup

- Database model

- ERD

- UML Diagram

- Sequence Diagram

- Class Diagram

- FLOW Diagram

- Scrum

- Unit Tests

- Beveiliging

- Prestatie overwegingen

- Installatie- en implementatiehandleiding

- Onderhoud en toekomstige ontwikkelingen

**Inleiding**

Welkom bij ons C# Pong-coderingsproject!

Een van de eerste en meest iconische videospellen, heeft generaties lang gamers vermaakt en is een hoeksteen geworden in de geschiedenis van de videogame-industrie. In dit project zullen we de klassieke Pong-game opnieuw creëren met behulp van de programmeertaal C#.

Pong staat bekend om zijn eenvoudige maar verslavende gameplay, waar twee spelers elk een peddel bedienen om een virtuele bal heen en weer te stuiteren. Het doel is eenvoudig: voorkomen dat de bal je kant van het scherm raakt, terwijl je probeert de bal terug te slaan naar je tegenstander om punten te scoren.

Door middel van dit project zullen we niet alleen de basisprincipes van C#-programmering verkennen, maar ook fundamentele concepten van game-ontwikkeling begrijpen, zoals het beheren van de spelstatus, het verwerken van gebruikersinvoer en het renderen van grafische elementen op het scherm.

Laten we samen duiken in de fascinerende wereld van Pong-ontwikkeling en onze eigen versie van dit tijdloze spel tot leven brengen!

**Versie Beheer**

* 1. Setup document. 8-2
  2. Wireframes en Mock ups. 14-2
  3. User story’s setup. 4-3

**Belangrijke rollen**

**The Product Owners**

Jan Zuur

Ron van Zuilichem

Tom Sievers

**SCRUM Master**

Joshua de Bruijn

**DEV Team**

Pedro Eduardo Cardoso

Joshua de Bruijn

Mitchel Meskes

**Plan van Eisen**

**Belangrijkste Kenmerken**

De applicatie zal gebruikmaken van object georiënteerd programmeren in C# om de code te structureren.

De belangrijkste functionaliteiten van de applicatie omvatten:

Gamemode kiezen, score weergave, winaar weergeven.

**Algemene doelen**

Het bouwen van een gebruiksvriendelijke Pong Game.

Het testen en valideren van de applicatie om ervoor te zorgen dat deze voldoet aan de gestelde eisen en normen.

**Huisstijl**

**Color pallet/ Font**

De kleurenpallet voor Pong zal uit originele kleuren bestaan:

**Gebruikte Kleuren:**

Player 1 bad: #0000FF (blauw)

Player 2 bad: #FF0000 (rood)

Pong Ball: #FFFFFF (wit)

Achtergrond:  #000000

Details: #FFFFFF

Font: Default font gebruikt.

Deze combinatie zorgt voor een rustig en netten omgeving voor de applicatie en het roept nostalgie op naar de al bestaande game.

**Buttons**

De knoppen zullen bestaan uit zwart en wit.

Achtergrondknop:  #000000

Detailsknop / text: #FFFFFF

**User story’s**

**Algemene Storys**

De user storys zijn gemaakt met behulp van Azure DevOps/Github Projects.

Hoe wij onze User story’s maken is:

We kiezen een EPIC delen die vervolgens op in een aantal Sprints en in een van die sprints pakken wij vervolgens delen van die EPIC op zodat ieder aan een deel van het project werkt.

**(Zie het volgende voorbeeld)**

**Stap 1**

Kies een EPIC. (Gebruikersregistratie en -authenticatie)

**Stap 2**

Vervolgens delen wij die in delen op en pakt ieder een story.

**Stap 3**

Dan kiest ieder een van de volgende punten en gaat daarmee aan de slag mee.

Iedere EPIC die we behandelen draaien we alle rollen een door zodat iedereen aan alle vlakken heeft gewerkt zowel backend, frontend als databases.

**User story’s**

**Epic 1: Maak Pong**

**User Stories for Frontend 1.1:**

**User Story 1.1.1:** Design Game Board Interface

As a player, I want to see a well-designed game board interface, so that I can easily understand the game layout.

**Acceptance Criteria:**

* + - Design a visually appealing game board with clear boundaries and markings for the playing area.
    - Ensure the game board layout is intuitive and easy to understand.

**User Story 1.1.2:** Implement Paddle Display

As a player, I want to see the paddles displayed on the game board, so that I can understand my position.

**Acceptance Criteria:**

* + - Display paddles as rectangular shapes on both sides of the game board.
    - Ensure paddles are positioned correctly and respond to player input for movement.

**User Story 1.1.3:** Display Ball

As a player, I want to see the ball moving on the game board, so that I can understand the game's dynamics.

**Acceptance Criteria:**

* + - Display the ball as a circular shape moving across the game board.
    - Ensure the ball moves smoothly and bounces off walls and paddles accurately.

**User story’s**

**User Story 1.1.4:** Score Display

As a player, I want to see my score displayed on the screen, so that I can track my progress.

**Acceptance Criteria:**

* + - Display the player's score prominently on the screen.
    - Update the score when a player scores a point by hitting the opponent's wall.

**Backlog Items for Frontend 1.1:**

**Backlog Item 1.1.1:** Implement Game Rendering Logic

As a front-end developer, I want to write code to render the game elements on the screen, so that players can see the game environment.

**Acceptance Criteria:**

* + - Use a graphics library or framework to render game elements such as the game board, paddles, ball, and score.
    - Ensure the game renders correctly on different screen sizes and resolutions.

**User story’s**

**User Stories for Backend 1.2:**

**User Story 1.2.1:** Manage Paddle Movement

As a developer, I want to enable paddle movement based on player input, so that players can control their paddles.

**Acceptance Criteria:**

* + - Capture player input from keyboard or controller to move paddles up and down.
    - Limit paddle movement within the boundaries of the game board.

**User Story 1.2.2:** Implement Ball Physics

As a developer, I want to define the behavior of the ball, including its movement and collisions, so that the game feels realistic.

**Acceptance Criteria:**

* + - Define ball movement using physics principles such as velocity, acceleration, and collision detection.
    - Ensure the ball reflects off walls and paddles accurately according to the laws of physics.

**User Story 1.2.3:** Detect Collisions

As a developer, I want to detect collisions between the ball and paddles, as well as with the game boundaries, so that I can handle game interactions.

**Acceptance Criteria:**

* + - Implement collision detection algorithms to detect when the ball collides with paddles or game boundaries.
    - Trigger appropriate actions such as bouncing the ball off surfaces or updating player scores.

**User story’s**

**Backlog Items for Backend 1.2:**

**Backlog Item 1.2.1:** Implement Game Mechanics

As a backend developer, I want to code the core game mechanics for Pong, including ball movement, collision detection, and scoring rules.

**Acceptance Criteria:**

* + - Write code to manage the game loop, including updating the game state and rendering frames.
    - Implement scoring logic to award points when the ball hits the opponent's wall.
    - Ensure game mechanics are implemented accurately and efficiently.

**Wireframe**

**De wireframe bestaat uit de volgende pagina’s:**

**Start Page**, Laat de Startpagina zien met een play button.

A white rectangle with black text

Description automatically generated

1. Is de box met de gamemode keuze en start.

**SinglePlayer Page**, Laat de Singleplayer pagina zien met de twee bads en een pingpongbal en de huidige score.

A white background with black text

Description automatically generated

1. Begin positie van de bal na score en het begin van de game.

2. Nummer twee zijn de bads en hun start positie

**Wireframe**

**SinglePlayer (na Score) Page**, Laat de Singleplayer pagina zien met de gescoorde punten na het raken van het tegenstanders goal.

A white background with black text

Description automatically generated

1. Na collisie met de achterwand zal de de pop up onstaan met wie heeft gescoord.

2. beweegbare bat.

3. Een tekstpop up met weergave van speler die gescoord heeft.

**Wireframe**

**SinglePlayer (na Win) Page**, Laat de Singleplayer pagina zien met de gescoorde punten en het win scherm met de speler die gewonnen heeft.

A black and white square with black text

Description automatically generated

1. na de derde collisie met de achterwand heeft een van de spelers gewonnen

2. plek van scores

3. Pop up in een box met de tekst met wie er gewonnen heeft.

**Mockup**

**Start Page**

A black screen with white text

Description automatically generated

**SinglePlayer Page**

A screen shot of a black background

Description automatically generated

**Mockup**

**SinglePlayer (na Score) Page**

A screen shot of a video game

Description automatically generated

**SinglePlayer (na Win) Page**

A screenshot of a video game

Description automatically generated

**Database model**

**ERD**

**UML Diagram**

**Sequence Diagram**

**Class Diagram**

**Flow Diagram**

**Scrum**

**Scrum Opzet**

Wij hebben de hele planning uitgewerkt met Azure DevOps/Github Projects zoals te zien is in de volgende kopjes.

En communicatie ging via Discord en Whatsapp.

**Way of Working:**

A screenshot of a computer

Description automatically generated

**Backlogbeheer:**

Aan het begin van de dag gaan we in Discord en houden wij onze dagelijkse stand-up vergaderingen en sprint review waar wij bespreken wat we gaan behandelen van de backlog, en delen wij die groteren EPICS op in meerdere sprints die vervolgens in meerdere User Storys omgezet worden.

A screenshot of a computer

Description automatically generated

**(Zie volgende pagina)**

**Scrum**

**Sprint Oplevering/ Sprintplanning:**

De groep levert de tot zover werkende software op aan het einde van elke sprint.

En melden vervolgens ook welke problemen zij hebben opgelopen met de opgepakte user story’s of welke ze allemaal afgemaakt hebben met succes.

**Scrum**

**Definition of Done:**

Onze D.O.D. Bestaat uit:

**Stap 1**

Het begint uiteraard met het schrijven van de code.

**Stap 2**

En dan moet de code gecommenten worden.

**Stap 3**

Het testen van die code in de aparte brench in zichzelf.

**Stap 4**

Het dan samenvoegen met de Main brench en het geheel testen.

**Stap 5**

Vervolgens meld je bij de groep dat je klaar bent en ga je verder met het volgende item op de sprint.

**Stap 6**

Aan het einde van die dag gaan we in onze afsluit stand up en gaan we over alle veranderingen en progressie die er gemaakt is zover en bespreken wij de zowel geschreven code als comments zodat we alles samen dubbelchecken.

En het dan allemaal definitief goedkeuren en als alles goed is sluiten wij die dag af.

Maar als wij tegen een probleem zijn gelopen bijvoorbeeld met een push conflict zullen wij elkaar in een call in discord gelijk alles verhelpen zodat iedereen productief blijft en weer verder kan.

**Unit Test**

Wij hebben ervoor gekozen om de unit tests tijdens onze productie te doen, een voorbeeld hiervan is:

**Beveiliging**

Project Setup:

**Prestatie Overweging**

**Installatie handleiding**

**Implementatie handleiding**

**Onderhoud**

**Toekomstige ontwikkelingen**