Tomas Heij, Djomar Lopulissa

Technisch ontwerp

Het project tot in detail uitgelegd

Inhoud

[1 Revisiehistorie 2](#_Toc209432448)

[Inleiding 3](#_Toc209432449)

[2 Doel 4](#_Toc209432450)

[1.1 Beoogde lezer 4](#_Toc209432451)

[1.2 Scope 4](#_Toc209432452)

[3 Software Architectuur 5](#_Toc209432453)

[1.3 Architectuuroverzicht 5](#_Toc209432454)

[1.4 Systeemvereisten 6](#_Toc209432455)

[1.5 Databaseverbinding 7](#_Toc209432456)

[4 Technische keuzes 8](#_Toc209432457)

[1.6 Technische uitdagingen 8](#_Toc209432458)

[1.7 Oplossingen voor uitdagingen 8](#_Toc209432459)

[1.8 Gekoze oplossing en verantwoording 8](#_Toc209432460)

[5 Database 9](#_Toc209432461)

[1.9 Entity-relationship-Diagram/ERD 9](#_Toc209432462)

[1.10 Tabel specificaties 9](#_Toc209432463)

[1.11 Data dictionary 10](#_Toc209432464)

[6 Opbouw van de applicatie 11](#_Toc209432465)

[1.12 Taal en Framework 11](#_Toc209432466)

[1.13 Diagrammen 11](#_Toc209432467)

[1.14 Klassendiagram 11](#_Toc209432468)

[1.15 Use-Case Diagrammen 12](#_Toc209432469)

[1.16 Activity Diagram 12](#_Toc209432470)

[1.17 Bibliotheken 13](#_Toc209432471)

[1.18 Codeafspraken 14](#_Toc209432472)

[7 Beveiligingsrichtlijnen 15](#_Toc209432473)

[1.19 OWASP top 10 15](#_Toc209432474)

[1.20 Beveiligingsmaatregelen 15](#_Toc209432475)

# Revisiehistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Auteur | Datum | Opmerkingen |
| V0.0.1 | T.H | 10-09-2025 | Basis opmaak gemaakt |
| V0.0.2 | T.H | 15-09-2025 | Basis opmaak verder opgemaakt |
|  |  |  |  |

# Inleiding

Dit is het technisch ontwerp voor ons sancties project. Hierin staat het technische van het project tot in detail uitgewerkt.

# Doel

Duidelijkheid geven over ons project.

## Beoogde lezer

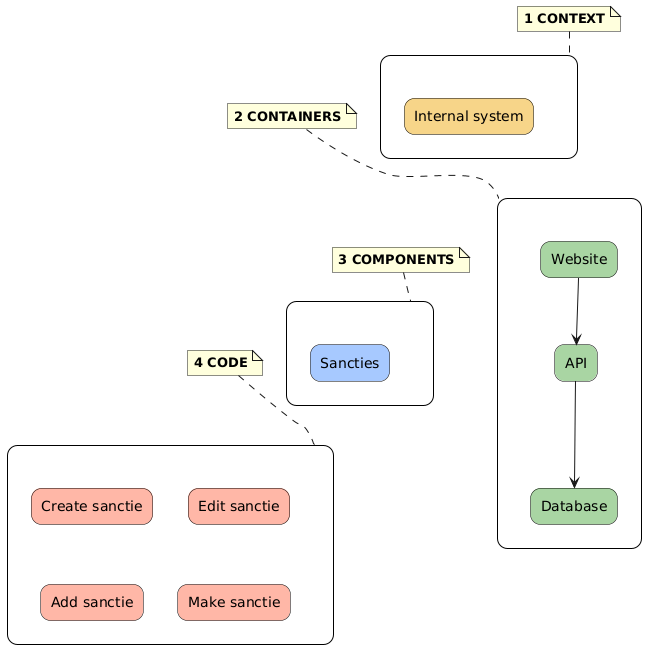
Beoordelaar van het project

## Scope

Geen integraties met externe applicaties

# Software Architectuur

## Architectuuroverzicht



## Systeemvereisten

Server:

* express v4.21.2 or higher
* path v0.12.7 or higher
* sqlite3 v5.1.7 or higher
* next: 15.5.2 or higher
* react-dom: 19.1.0 or higher
* react: 19.1.0 or higher

Client:

* Moderne webbrowser (Chrome, Firefox, Edge of Safari)
* Minimale resolutie desktop: 1024x768
* Minimale resolute telefoon: 360 x 640

Hardware:

* Processor: 1 GHz, dual-core
* Geheugen: 2 GB RAM
* Opslag: 2 GB vrije opslagruimte
* Grafische kaart: Geïntegreerde GPU
* Netwerk: 1 Mbps download/upload
* Besturingssysteem: Windows 7/macOS 10.12/Linux (moderne distros)
* Mobiel: Android 5.0/iOS 12.0 of hoger

## Databaseverbinding

Verbinding gaat met behulp van een fetch request binnen de client naar een API in de server waarin de verbinding met de database wordt gelegd. Er zijn verschillende API endpoints voor verschillende doeleinden zoals sancties aanmaken, bewerken en toevoegen.

const db = new sqlite3.Database('database.sqlite');

db.serialize();

# Technische keuzes

## Technische uitdagingen

* SQL Injecties bij het toevoegen van sancties
* SQLite kan niet over GIT

## Oplossingen voor uitdagingen

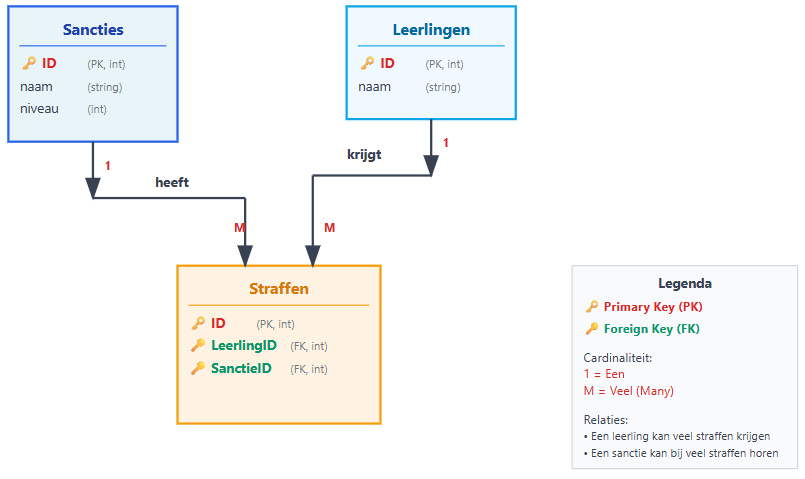
* SQL injectie: door SQL laten controleren voordat je het uitvoert
* Database niet over GIT versturen maar i.p.v. dat seeden

## Gekoze oplossing en verantwoording

Het is eenvoudig en voor de hand liggend. Ze voldoen aan de eisen en dus is het niet nodig om verder te kijken.

# Database

## Entity-relationship-Diagram/ERD



## Tabel specificaties

**Tabel: Sancties**

* PK: ID – uniek ID van de sanctie
* FK: geen

**Tabel: Leerlingen**

* PK: ID – uniek ID van de leerling
* FK: geen

**Tabel: Straffen**

* PK: ID – uniek ID van de straf
* FK: LeerlingID → Leerlingen.ID
* FK: SanctieID → Sancties.ID

## Data dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Tabelnaam | Kolomnaam | Type | Sleutel |
| Sancties | ID | int | PK |
|  | naam | string |  |
|  | niveau | int |  |
|  |  |  |  |
| Leerlingen | ID | int | PK |
|  | naam | string |  |
|  |  |  |  |
| Straffen | ID | int | PK |
|  | Leerling ID | int | FK |
|  | Sanctie ID | Int | FK |
|  |  |  |  |

# Opbouw van de applicatie

## Taal en Framework

De applicatie maakt gebruik van verschillende technologieën met elk hun eigen functie. Node JS regelt de server en Next JS, Typescript en Bootstrap regelen de website. Voor de opslag van gegevens gebruiken wij Prisma.

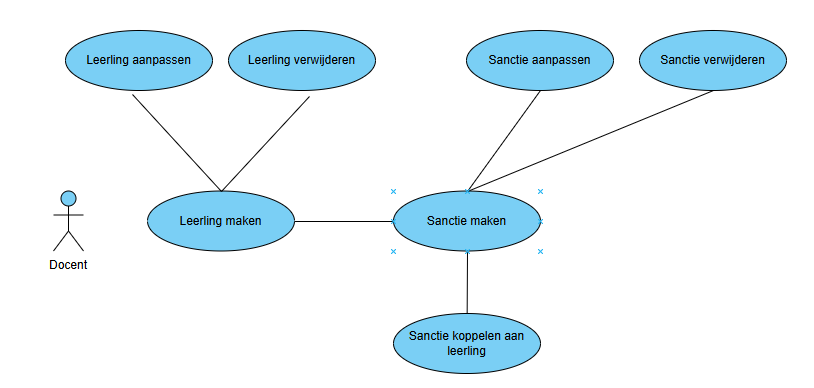
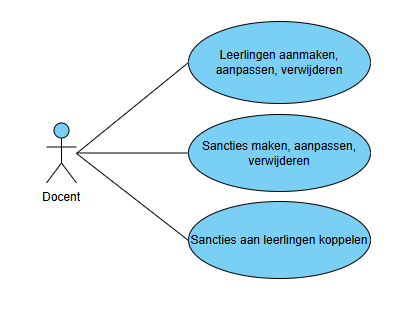
## Diagrammen

Zijn hieronder vermeld

## Klassendiagram

..

## Use-Case Diagrammen



## Activity Diagram

..

## Bibliotheken

**Backend**

- Node.js met Next JS

- Prisma database

**Frontend**

- React (Next JS)

**Database**

- Prisma lokaal

- Tabellen: Leerlingen, Sancties, Straffen

**Server**

- Port 3000

- API endpoints: /leerlingen, /sancties, /straffen

**Beveiliging**

- N.v.t

## Codeafspraken

**Naamgeving**

* Cconstanten: camelCase
* Functies: PascalCase

**Imports**

* External eerst
* Internal daarna (ex: APIs)

**Comments**

* Boven elke functie met jouw naam en wat het doet

**Auth Pattern**

* N.v.t

**Debug**

* Toggle: const debug = false
* Conditional: if (debug) console.log(...)

**Routes**

* Geen leerling: redirect naar leerling maken
* Sanctie toevoegen: redirect naar sanctie maken

# Beveiligingsrichtlijnen

## OWASP top 10

* Injection.
* Security Misconfiguration.
* Vulnerable and Outdated Components.

## Beveiligingsmaatregelen

* SQL strings worden gesanitiseerd voordat ze worden uitgevord.
* de server weergeeft minimale errors en toont nooit meer dan nodig
* al onze libraries gebruiken hun laatste versie