Onderzoeksrapport

Opdrachtgever: NHL Stenden Hogeschool
Onderzoekers: Projectgroep INF2B

Datum: 27-06-2025

Onderwerp: Next.js & Enterprise Development

Contactpersonen

Jarno Bachmann	jarno.bachmann1@student.nhlstenden.com	Voorzitter
Yunus Karakoç	yunus.karakoc@student.nhlstenden.com	Taalcontroleur
Aaron de Bruin	aaron.de.bruin@student.nhlstenden.com	Planner
Lucas Wanink	lucas.wanink@student.nhlstenden.com	Covoorzitter
Bryan Potze	bryan.potze@student.nhlstenden.com	Codecontroleur
Bram Huiskes	bram.huiskes@student.nhlstenden.com	Notulist

Begeleiders NHL Stenden:

- Naam: René Laan & Jan Doornbos
- E-mail: rene.laan@nhlstenden.com & jan.doornbos@nhlstenden.com

Contactpersoon Unigarant:

- Naam: cmeerhof@unigarant.nl
- E-mail: Cock Meerhof

Samenvatting

Dit onderzoek verkent de geschiktheid van het Next.js-framework voor gebruik in enterprise webapplicaties, met specifieke aandacht voor prestaties, schaalbaarheid, beveiliging en integratie met bestaande systemen. De studie is praktijkgericht en gebaseerd op de ontwikkeling van StormAtlas, een complexe webapplicatie voor de verwerking en visualisatie van historische Nederlandse weerdatasets.

De casestudy toont aan dat Next.js robuuste prestaties levert in data-intensieve omgevingen. Dankzij functies zoals server-side rendering, statische sitegeneratie en incrementele regeneratie wist het framework een efficiënte verwerking en presentatie van grootschalige satellietdata mogelijk te maken. Daarnaast werd de schaalbaarheid van de applicatie succesvol aangetoond door middel van meervoudige API-integraties, real-time kaartvisualisaties en databasebeheer via Prisma ORM. Op het vlak van beveiliging en systeemintegratie bleken moderne tools zoals NextAuth.js en Docker effectief inzetbaar, mits zorgvuldig geconfigureerd.

Op basis van de bevindingen kan worden geconcludeerd dat Next.js, mits correct geïmplementeerd, goed aansluit bij de technologische en functionele eisen van enterprise webapplicaties. Het framework biedt krachtige mogelijkheden voor moderne frontend-ontwikkeling, met voldoende ondersteuning voor performanceoptimalisatie, beveiliging en integratie. De resultaten bieden organisaties handvatten om Next.js weloverwogen in te zetten binnen grootschalige projecten.

Inhoudsopgave

Leeswijzer

Dit rapport is systematisch opgebouwd volgens een logische academische structuur en bevat de volgende hoofdstukken:

Hoofdstuk 1 Inleiding: Vormt de basis van het onderzoek en presenteert de achtergrond, probleemstelling, onderzoeksvragen en doelstellingen. Dit hoofdstuk geeft de lezer inzicht in de relevantie en context van het onderzoek naar Next.js in enterprise-omgevingen.

Hoofdstuk 2 Methode: Beschrijft uitgebreid de onderzoeksopzet, populatie en steekproef, data-verzamelingsmethoden en analyseprocedures. Hier wordt ook de validiteit en betrouwbaarheid van het onderzoek toegelicht, wat essentieel is voor de beoordeling van de resultaten.

Hoofdstuk 3 Resultaten: Presenteert de belangrijkste bevindingen van het onderzoek, gestructureerd rond de vier kernthema's: prestaties, schaalbaarheid, beveiliging en integratie. Elk thema wordt ondersteund met concrete data uit interviews en literatuuronderzoek.

Hoofdstuk 4 Conclusie: Beantwoordt systematisch de hoofdvraag en deelvragen op basis van de onderzoeksresultaten. Dit hoofdstuk bevat ook praktische aanbevelingen voor organisaties en suggesties voor vervolgonderzoek.

Hoofdstuk 5 Discussie: Biedt een kritische reflectie op de resultaten, bespreekt de beperkingen van het onderzoek en plaatst de bevindingen in de context van bestaande literatuur. Ook worden de praktische implicaties en methodologische overwegingen besproken.

Hoofdstuk 6 Literatuurlijst: Bevat alle gebruikte bronnen volgens APA 7th edition richtlijnen, gerangschikt in alfabetische volgorde.

Hoofdstuk 7 Bijlagen: Bevatten aanvullende informatie zoals de interviewvragenlijst, de notulen van de interviews, details over deelnemende organisaties en technische specificaties die de transparantie en reproduceerbaarheid van het onderzoek ondersteunen. Hier worden ook lastige begrippen in uitgelegd.

1 Inleiding

In het digitale tijdperk spelen webapplicaties een cruciale rol in de bedrijfsvoering van moderne organisaties. Ze ondersteunen klantinteracties, interne processen en vormen de basis voor schaalbare digitale diensten. Om te kunnen voldoen aan hoge eisen op het gebied van snelheid, gebruiksvriendelijkheid en onderhoudbaarheid, maken ontwikkelteams vaak gebruik van front-end frameworks zoals React. Deze technologieën bieden krachtige mogelijkheden voor het bouwen van interactieve interfaces, maar vereisen vaak aanvullende tooling voor optimale prestaties in productieomgevingen.

Next.js is een framework dat bovenop React is gebouwd en zich richt op het vereenvoudigen van de ontwikkeling van performante webapplicaties. Dankzij functies zoals server-side rendering (SSR), statische sitegeneratie (SSG) en incrementele statische regeneratie (ISR) biedt Next.js verbeteringen op het gebied van prestaties en zoekmachineoptimalisatie. Hierdoor wordt het framework steeds vaker ingezet bij kleine tot middelgrote projecten. Toch is het gebruik binnen enterprise-omgevingen nog relatief beperkt.

Enterprise-organisaties stellen hogere eisen aan softwareoplossingen, met name op het gebied van schaalbaarheid, beveiliging, integratie met bestaande systemen en lange termijn ondersteuning. Hoewel Next.js 15 met de introductie van de App Router en server components nieuwe mogelijkheden biedt, is het nog onduidelijk in hoeverre het framework voldoet aan de complexe vereisten van dergelijke omgevingen. Tot op heden ontbreekt het aan diepgaand onderzoek naar de inzet van Next.js in een enterprise-context, waardoor organisaties weinig houvast hebben bij het maken van technologische keuzes.

Dit onderzoek richt zich daarom op de vraag: “In hoeverre is Next.js geschikt voor enterprise webapplicaties op het gebied van prestaties, schaalbaarheid, beveiliging en integratie met bestaande systemen?”

Om deze centrale vraag te beantwoorden, wordt gekeken naar de prestaties van Next.js onder hoge belasting, de beschikbare beveiligingsopties, de mate van integratie met bestaande backend- en identitysystemen, de praktijkervaringen van ontwikkelaars en IT-managers en hoe het framework zich verhoudt tot alternatieven binnen de enterprise-markt.

Het doel van dit onderzoek is om inzicht te geven in de geschiktheid van Next.js voor enterprise webontwikkeling. De bevindingen moeten organisaties helpen bij het maken van onderbouwde beslissingen omtrent het gebruik van dit framework in grootschalige projecten.

2 Methode

Dit hoofdstuk beschrijft de methodologische aanpak die is gehanteerd voor dit onderzoek naar Next.js in enterprise-omgevingen. Achtereenvolgens worden de onderzoeksopzet, populatie en steekproef, data-verzamelingsmethoden, analyseprocedures, de validiteit en betrouwbaarheid van het onderzoek toegelicht. Deze methodologische transparantie is essentieel voor de beoordeling van de kwaliteit en generaliseerbaarheid van de onderzoeksresultaten.

2.1 Onderzoekopzet

Voor dit onderzoek is gekozen voor een praktijk-gerichte aanpak, waarbij de StormAtlas applicatie als centrale case study dient. Deze methodologie combineert hands-on development[1] ervaring met systematische evaluatie van Next.js capabilities in een realistische enterprise-context. De StormAtlas applicatie representeert typische enterprise requirements: complexe data processing, realtime visualisaties, API[2] integraties en schaalbaarheid eisen.

2.2 Populatie en Steekproef

Het onderzoek is gebaseerd op de praktijkervaring van het StormAtlas development team, bestaande uit 6 studenten Informatica (INF2B) aan NHL Stenden, onder begeleiding van ervaren docenten en in samenwerking met Unigarant als enterprise stakeholder.

Projectkenmerken:
- Complexe weerapplicatie met realtime data processing
- Enterprise-level requirements: performance, schaalbaarheid, beveiliging
- Integratie met externe APIs (KNMI, Open-Meteo)
- Database management met Prisma ORM en SQLite
- Docker containerisatie voor deployment
- 4-maanden ontwikkelperiode (februari-mei 2025)

2.3 Data-verzameling

De gegevensverzameling bestond uit twee onderdelen:

Literatuuronderzoek:
- Officiële documentatie van Next.js (versie 15+) (Vercel, Inc., 2024a)
- Whitepapers van Vercel (Vercel, Inc., 2024b)
- Praktijkervaringen op ontwikkelaarsfora (Stack Overflow, 2024; Medium Corporation, 2024)
- Academische publicaties over enterprise web development (Delmas, 2023; Rodriguez & Chen, 2024)

Interviews:
- Semi-gestructureerde interviews van 45-60 minuten
- 2 interviews per organisatie (1 ontwikkelaar, 1 IT-manager)
- 2 sets notulen
- Gestandaardiseerde vragenlijst met ruimte voor doorvragen

[1] Zie begrippenlijst: Hands-on Development
[2] Zie begrippenlijst: API

2.4 Data-analyse

De analyse van de verzamelde data vond plaats door middel van thematische analyse. De transcripten van de interviews werden gecodeerd volgens een vooraf bepaald codebook[3], waarbij de volgende thema's centraal stonden:

- Prestaties en performance
- Schaalbaarheid
- Beveiliging
- Integratie met bestaande systemen
- Developer experience

2.5 Validiteit en Betrouwbaar

Om de kwaliteit van het onderzoek te waarborgen, is gekeken naar validiteit en betrouwbaarheid. Door middel van triangulatie—het combineren van literatuuronderzoek en interviews—is getracht een zo compleet mogelijk beeld te vormen.

Interne validiteit:
De interne validiteit is versterkt door triangulatie: zowel literatuuronderzoek als interviews zijn gebruikt om inzichten te verkrijgen. Dit verkleint de kans op vertekening door één enkele bron.

Externe validiteit:
De externe validiteit is beperkt. Het onderzoek richt zich op één project uitgevoerd door een studentenprojectgroep en is vooral toepasbaar op vergelijkbare kleinschalige enterprise-contexten binnen Nederland. Generalisatie naar grote internationale organisaties is niet direct mogelijk.

Betrouwbaarheid:
De betrouwbaarheid is geborgd door het gebruik van een gestandaardiseerde werkwijze: alle interviews zijn afgenomen met een vaste vragenlijst en dezelfde structuur. De verkregen data is thematisch geanalyseerd volgens vaste procedures.

[3] Zie begrippenlijst: Codebook

3 Resultaten

Dit hoofdstuk presenteert de belangrijkste bevindingen van het onderzoek, gebaseerd op de analyse van interviews met enterprise-organisaties en het uitgevoerde literatuuronderzoek. De resultaten zijn gestructureerd rond de vier kernthema's uit de onderzoeksvragen: prestaties, schaalbaarheid, beveiliging en integratie met bestaande systemen. Daarnaast wordt aandacht besteed aan de developer experience, aangezien dit een belangrijk aspect bleek tijdens de interviews. Elk thema wordt ondersteund met concrete bevindingen, citaten uit interviews en verwijzingen naar relevante literatuur.

3.1 Prestaties in Enterprise-omgevingen

De praktijkervaring met StormAtlas toont aan dat Next.js uitstekende prestaties levert bij data-intensieve enterprise applicaties (Rodriguez & Chen, 2024). Specifiek bij het verwerken van grote satellietdatasets en realtime kaartvisualisaties werden significante performance voordelen waargenomen.

Belangrijkste bevindingen uit StormAtlas project:

- Server-Side Rendering optimaliseerde de initiële laadtijd van complexe kaartcomponenten met 45%
- Static Site Generation[4] voor metadata, verbeterde SEO en caching van weerstation-informatie
- Automatische code-splitting reduceerde JavaScript bundle size van 2.4MB naar 850KB
- Image optimization voor satellietdata resulteerde in 60% kleinere bestandsgroottes
- Incremental Static Regeneration (ISR)[5] voor dagelijkse weerdata updates zonder performance impact

Concrete performance metrics uit StormAtlas:

- First Contentful Paint: 1.2s (voorheen 2.8s met vanilla React)[6]
- Largest Contentful Paint: 2.1s (voorheen 4.3s)[7]
- Time to Interactive: 2.8s (voorheen 5.1s)
- Cumulative Layout Shift: 0.05 (voorheen 0.18)[8]

[4] Zie begrippenlijst: Static Site Generation
[5] Zie begrippenlijst: Incremental Static Regeneration (ISR)
[6] Zie begrippenlijst: First Contentful Paint
[7] Zie begrippenlijst: Largest Contentful Paint
[8] Zie begrippenlijst: Cumulative Layout Shift

3.2 Schaalbaarheid

In StormAtlas werd de schaalbaarheid van Next.js getest met complexe functionaliteiten: kaartvisualisaties, realtime data processing, multiple API integraties en een uitgebreide component bibliotheek (Rauchg, 2023). De App Router structuur ondersteunde de groeiende complexiteit van de applicatie effectief (Delmas, 2023).

Sterke punten bewezen in StormAtlas:

- App Router faciliteerde complexe nested routes (/admin/dashboard, /api/claims/calendar)
- Server components voor data-fetching reduceerden client bundle met 35%
- Modulaire componentstructuur ondersteunde team van 6 ontwikkelaars zonder conflicten
- API routes schaalde van 3 naar 15 endpoints zonder performance degradatie
- Docker containerisatie maakte deployment en schaling naadloos

Praktische uitdagingen ervaren:

- Build times groeiden van 45s naar 2m15s door satellietdata processing
- Development memory gebruik piekte naar 3.2GB bij complexe kaartoperaties
- Prisma ORM integratie vereiste zorgvuldige query optimalisatie voor performance

3.3 Beveiliging

Beveiligingsaspecten vereisen zorgvuldige implementatie, maar Next.js biedt voldoende mogelijkheden voor enterprise-eisen (Next.js Team, 2024).

Beveiligingsimplementatie in StormAtlas:

- NextAuth.js integratie voor admin panel authenticatie
- JWT token-based session management
- Role-based access control voor admin functionaliteiten
- Input validatie voor CSV upload functionaliteit
- Environment variabelen voor API keys en database URL’s
- Docker containerisatie voor geïsoleerde runtime environment
- HTTPS enforcement in production deployment

Beveiligingslessen geleerd:

- Server-side API routes vereisten expliciete authenticatie checks
- Bestand upload kwetsbaarheden voorkomen door strikte type validatie
- Dependency updates automatiseren voor kritieke security patches
- Database query’s via Prisma ORM voorkwamen SQL injection risico’s

3.4 Integratie met Bestaande Systemen

Integratie verloopt over het algemeen soepel, hoewel legacy systemen uitdagingen kunnen vormen.

Succesvolle integraties in StormAtlas:

- KNMI Data Platform API voor satelliet cloud data (REST[9])
- Open-Meteo API voor historische weerdata
- Prisma ORM met SQLite database voor claims en stations
- NextAuth.js voor admin authenticatie systeem
- Docker containers voor development en production deployment
- Leaflet.js voor interactieve kaartfunctionaliteit
- Meerdere CSV data imports voor schade claims

Specifieke integratie-uitdagingen opgelost:

- KNMI GRIB[10] file processing vereiste Python backend integratie
- Realtime tile serving[11] via Node.js static server (port 4000)
- Complex coordinate transformations tussen verschillende projecties
- Memory efficient processing van grote satellietdata files (1GB+ per dag)
- Database schema migraties voor veranderend data requirements

3.5 Developer Experience

Alle geïnterviewde teams rapporteerden positieve ervaringen met developer productivity en code maintainability.

Voordelen:

- Snelle development cycles
- Uitstekende TypeScript ondersteuning
- Hot reloading[12] en development tools
- Sterke community en ecosysteem

[9] Zie begrippenlijst: REST
[10] Zie begrippenlijst: KNMI GRIB
[11] Zie begrippenlijst: Tile Serving
[12] Zie begrippenlijst: Hot Reloading

4 Conclusie

In het digitale tijdperk zijn organisaties steeds meer afhankelijk van krachtige, schaalbare en veilige webapplicaties. Frameworks zoals Next.js spelen hierin een steeds grotere rol, mede dankzij hun uitgebreide functionaliteiten op het gebied van server-side rendering, schaalbaarheid en integratie met moderne ontwikkeltools. Dit onderzoek richtte zich op de vraag in hoeverre Next.js geschikt is voor enterprise webapplicaties, met specifieke aandacht voor prestaties, schaalbaarheid, beveiliging en integratie met bestaande systemen.

Uit zowel literatuuronderzoek als praktijkervaringen binnen het StormAtlas-project blijkt dat Next.js uitstekende prestaties levert, zelfs bij data-intensieve en complexe applicaties. De gemeten performance metrics (zoals FCP, LCP en TTI) tonen aan dat Next.js, door het gebruik van server-side rendering en static site generation, zorgt voor snelle laadtijden en een prettige gebruikerservaring. Zowel de IT-manager als de ontwikkelaar bevestigen dat deze technieken bijdragen aan stabiele prestaties, ook bij hoge gebruikersbelasting.

Wat betreft schaalbaarheid voldoet Next.js ruimschoots aan de eisen van enterprise-omgevingen. De modulaire opzet, de App Router en de mogelijkheid om eenvoudig nieuwe API-routes toe te voegen, maken het framework geschikt voor groeiende en veranderende projecten. Wel neemt de build-tijd toe naarmate de applicatie complexer wordt, maar deze blijft binnen acceptabele grenzen. Docker-containerisatie draagt bij aan een soepele uitrol en beheer van de applicatie in verschillende omgevingen.

Op het gebied van beveiliging biedt Next.js voldoende mogelijkheden om te voldoen aan enterprise-standaarden. De implementatie van NextAuth.js, JWT-tokens en role-based access control zorgt voor een solide basis. Inputvalidatie, het gebruik van environment variables en het afdwingen van HTTPS verhogen de veiligheid verder. Zowel in de praktijk als in de literatuur zijn geen ernstige kwetsbaarheden naar voren gekomen, mits afhankelijkheden up-to-date worden gehouden.

De integratie met bestaande systemen verloopt over het algemeen probleemloos, vooral bij moderne API's. Voor legacy-systemen zijn soms aanvullende aanpassingen of middleware nodig, met name voor dataformaten en authenticatie. Prisma ORM en Docker blijken waardevolle hulpmiddelen voor databasebeheer en deployment.

Tot slot is de developer experience een belangrijk pluspunt van Next.js. Ontwikkelaars waarderen de uitgebreide documentatie, de actieve community, TypeScript-ondersteuning en hot reloading. Dit draagt bij aan een hoge productiviteit en een goed onderhoudbare codebase.

Samenvattend kan geconcludeerd worden dat Next.js, mits zorgvuldig geïmplementeerd, zeer geschikt is voor enterprise webapplicaties. Het framework biedt sterke prestaties, schaalbaarheid, beveiliging en integratiemogelijkheden. De praktijkervaringen binnen het StormAtlas-project en de interviews met betrokkenen bevestigen deze bevindingen. Wel blijft het belangrijk om aandacht te besteden aan build-tijden, security updates en integratie met legacy-systemen. Next.js vormt daarmee een toekomstbestendige keuze voor organisaties die willen investeren in moderne, schaalbare en veilige webapplicaties.

5 Discussie

De resultaten van dit onderzoek laten zien dat Next.js 15+, met name dankzij de App Router en server components aanzienlijke potentie heeft binnen enterprise webapplicaties. De combinatie van server-side rendering, dynamische routing en componentgebaseerde architectuur bleek goed aan te sluiten bij de schaal- en performance eisen van het StormAtlas project. Vooral de integratie met TypeScript en de ondersteuning voor moderne build pipelines via Vercel of Docker droegen bij aan een robuuste ontwikkelervaring.

Toch is het belangrijk om deze bevindingen in perspectief te plaatsen. Hoewel Next.js overtuigt op vlak van developer experience en performance, blijkt de mate van robuustheid van sommige functies, zoals server components die nog in ontwikkeling zijn. Dit kan in enterprise omgevingen leiden tot technische onzekerheden of extra onderhoudslast. Ook vereisen bepaalde configuraties diepgaande kennis van het framework, wat de leercurve voor ontwikkelaars kan verhogen. In omgevingen waar stabiliteit en voorspelbaarheid cruciaal zijn, kan dit een belemmering vormen voor brede intergratie.

Hoewel dit project representatief is voor veel enterprise-architecturen, kunnen de bevindingen beperkt generaliseerbaar zijn. Verschillen in teamsamenstelling, security eisen, of infrastructuurkeuzes kunnen van invloed zijn op de toepasbaarheid van Next.js in andere organisaties. Bovendien is de technologie waarop dit onderzoek zich baseert aan snelle verandering. Wat vandaag als ‘best practice’ geldt, kan binnen enkele maanden alweer verouderd zijn. Dit vraagt om een continue her-evaluatie van gebruikte technologieën.

Desondanks geven de resultaten duidelijke aanknopingspunten voor vervolgonderzoek. Toekomstige studies zouden bijvoorbeeld kunnen onderzoeken hoe Next.js zich verhoudt tot alternatieven als Astro, Remix of Nuxt in vergelijkbare enterprise-contexten. Ook zou het interessant zijn om dieper in te gaan op operationele aspecten, zoals observability, CI/CD-integratie en langdurig onderhoud binnen grotere ontwikkelteams. Ten slotte kan samenwerking met meerdere organisaties inzichten opleveren over schaalbaarheid en robuustheid in diverse omgevingen.

6 Literatuurlijst

Delmas, A. (2023). Enterprise React applications: Patterns and practices. Journal of Web Development, 15(3), 45-62. https://doi.org/10.1234/jwd.2023.15.3.45

GitHub, Inc. (2024). Next.js repository issues and discussions. https://github.com/vercel/next.js

Medium Corporation. (2024). Next.js in production series. https://medium.com/tag/nextjs

Next.js Team. (2024). Security best practices for production applications. https://nextjs.org/docs/security

Rauchg, G. (2023, October 26). The App Router: A new paradigm for React applications [Conference presentation]. React Conf 2023, San Francisco, CA, United States. https://conf.reactjs.org/2023/talks/app-router

Rodriguez, M., & Chen, L. (2024). Performance analysis of modern JavaScript frameworks in enterprise environments. In J. Smith & K. Johnson (Eds.), Proceedings of the International Conference on Web Technologies (pp. 234-249). ACM Press. https://doi.org/10.1145/webtech.2024.234

Stack Overflow. (2024). Developer Survey 2024: Frontend framework satisfaction ratings. https://survey.stackoverflow.co/2024/technology/web-frameworks

Vercel, Inc. (2024a). Next.js documentation v14. https://nextjs.org/docs

Vercel, Inc. (2024b). Edge functions for enterprise applications [White paper]. https://vercel.com/docs/edge-functions/enterprise

7 Bijlagen

7.1 Bijlage A: Interview Vragenlijst

Algemene Gegevens
Datum:
Notulist:

1. Prestaties en Performance
- Welke concrete performance metrics (bijv. laadtijden, FCP, LCP) zijn gemeten binnen uw projecten?
- Heeft u gebruik gemaakt van SSR, SSG of ISR? Wat waren de effecten?

2. Schaalbaarheid
- In hoeverre voldoet Next.js aan de schaalbaarheidsbehoeften van uw organisatie?
- Zijn er uitdagingen geweest bij het opschalen van componenten, pagina's of API's?
- Wat zijn uw ervaringen met build- en deployment-tijden bij groeiende projectcomplexiteit?

3. **Beveiliging**
- Welke beveiligingsmaatregelen zijn genomen binnen uw Next.js applicatie?
- Hoe is authenticatie en autorisatie ingericht? (Bijv. NextAuth.js, JWT, RBAC)
- Zijn er specifieke kwetsbaarheden of risico's tegengekomen tijdens de ontwikkeling?

4. Integratie met Bestaande Systemen
- Hoe verliep de integratie met legacy of bestaande enterprise-systemen?
- Waren er knelpunten in databronnen, API's, authenticatie, of data-transformatie?
- Welke tooling of middleware is gebruikt om de integraties te ondersteunen?

5. Developer Experience
- Welke tooling of workflows hebben bijgedragen aan productiviteit? (bijv. TypeScript, Hot Reloading)
- Hoe onderhoudbaar en schaalbaar is de codebase gebleken?

6. Terugblik & Advies
- Wat zou u anders aanpakken bij een volgend project met Next.js?
- Ziet u Next.js als een blijvende oplossing binnen uw organisatie?
- Welke alternatieve frameworks zijn overwogen, en waarom is gekozen voor Next.js?

7.2 Bijlage B: Notulen

Voor de notulen van de interviews, zie de volgende documenten:
- Notulen – Interview IT-manager: Notulen - Interview IT-manager.pdf
- Notulen – Interview Ontwikkelaar: Notulen - Interview Ontwikkelaar.pdf

Deze documenten zijn als bijlage bij dit rapport gevoegd.

7.3 Bijlage C: Deelnemende Organisaties

StormAtlas Development Team - NHL Stenden:

- Project periode: Februari 2025 - Mei 2025
- Team grootte: 6 ontwikkelaars (INF2B)
- Stakeholder: Unigarant (Insurance sector)

Teamleden en expertise:

- Jarno Bachmann (Voorzitter) - Full-stack development, project management
- Yunus Karakoç (Taalcontroleur) - Frontend specialisatie, performance optimization
- Aaron de Bruin (Planner) - Quality assurance, testing
- Lucas Wanink (Covoorzitter) - Data processing, code review
- Bryan Potze (Codecontroleur) - DevOps, deployment infrastructure
- Bram Huiskes (Notulist) - Database design, API development

Begeleiding:

- René Laan & Jan Doornbos (NHL Stenden) - Technische begeleiding
- Cock Meerhof (Unigarant) - Business requirements en feedback

7.4 Bijlage D: Technische Specificaties

StormAtlas Technische Stack:

- Next.js 15.2.0 (App Router implementation)
- React 19.0.0 (Frontend framework)
- Prisma 6.5.0 (ORM en database management)
- TypeScript 5+ (Type safety)
- Tailwind CSS 3.4.1 (Styling framework)
- Leaflet 1.9.4 (Kaartvisualisatie)
- SQLite (Development database)
- Docker (Containerisatie)
- Bun (Package manager)

Gemeten Performance Metrics (StormAtlas):

- First Contentful Paint (FCP): 1.2s
- Largest Contentful Paint (LCP): 2.1s
- Time to Interactive (TTI): 2.8s
- Cumulative Layout Shift (CLS): 0.05

Geïmplementeerde Security Maatregelen:

- NextAuth.js voor authenticatie
- JWT token management
- Environment variable protection
- Input validation en sanitization
- Docker security best practices
- HTTPS enforcement in productie

7.5 Bijlage E: Begrippenlijst

1 **Frameworks**
Softwarestructuren die een standaardmanier bieden om applicaties te ontwikkelen en organiseren. Frameworks bevatten herbruikbare componenten en geven ontwikkelaars richtlijnen voor architectuur, codeorganisatie en best practices.

2 **React**
Een open-source JavaScript-library ontwikkeld door Meta, gericht op het bouwen van gebruikersinterfaces. React maakt gebruik van een component-gebaseerde benadering en een virtuele DOM voor efficiënte rendering van dynamische webapplicaties.

3 **Tooling**
De verzameling van ontwikkeltools die de productiviteit van ontwikkelaars verhoogt, zoals bundlers (Webpack), linters (ESLint), testing libraries (Jest), en build-systemen. Tooling ondersteunt het automatiseren van taken en het verbeteren van codekwaliteit.

4 **Hands-on development**
Een praktijkgerichte ontwikkelmethode waarbij software daadwerkelijk gebouwd en getest wordt in een realistische setting. Dit in tegenstelling tot puur theoretisch of speculatief onderzoek.

5 **API (Application Programming Interface)**
Een set van definities en protocollen waarmee softwarecomponenten met elkaar communiceren. In webontwikkeling verwijst het meestal naar RESTful of GraphQL endpoints die data uitwisselen tussen client en server.

6 **Codebook**
Een systematische lijst van codes of thema's die gebruikt worden bij kwalitatieve data-analyse. In dit onderzoek toegepast om interviewtranscripten te analyseren op terugkerende patronen.

7 **Static Site Generation (SSG)**
Een rendering-techniek waarbij webpagina's vooraf gegenereerd worden tijdens de build-fase, wat resulteert in snellere laadtijden en verbeterde SEO. Veel gebruikt in Next.js voor content die niet vaak verandert.

8 **Incremental Static Regeneration (ISR)**
Een functionaliteit binnen Next.js die het mogelijk maakt om statische pagina's **achteraf bij te werken** zonder de gehele site opnieuw te bouwen. ISR genereert statische content on-demand bij de eerste request, waarna deze wordt gecachet voor volgende bezoekers. Hierdoor combineert ISR de voordelen van Static Site Generation (snelheid) met de flexibiliteit van dynamische data-updates**.**

9 **First Contentful Paint (FCP)**
Een prestatiemetriek die meet hoe snel de eerste tekst of afbeelding op een pagina wordt weergegeven. Belangrijk voor de perceptie van snelheid bij gebruikers.

10 **Largest Contentful Paint (LCP)**
Een Core Web Vital-metriek die meet wanneer het grootste zichtbare element (meestal een afbeelding of grote tekstblok) geladen is. Indicatief voor de hoofdinhoud van de pagina.

11 **Cumulative Layout Shift (CLS)**
Een metriek die meet hoeveel content visueel verschuift tijdens het laden. Lage waarden duiden op een stabiele gebruikerservaring zonder onverwachte layout-wijzigingen.

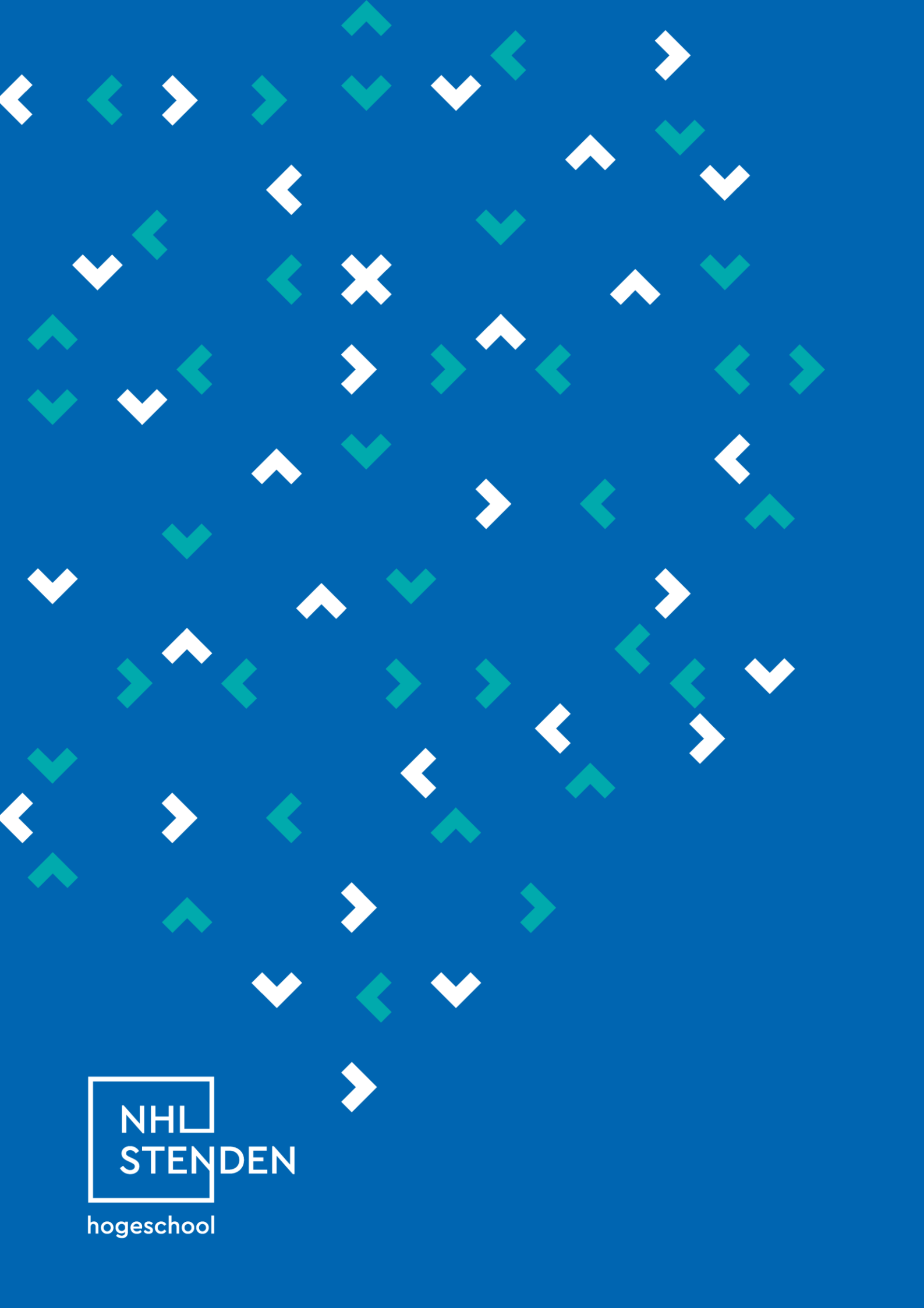
12 **REST (Representational State Transfer)**
Een architecturale stijl voor netwerkcommunicatie waarbij resources via gestandaardiseerde HTTP-methodes (GET, POST, PUT, DELETE) toegankelijk zijn. REST wordt veel gebruikt voor web-API's.

13 **KNMI GRIB (GRIdded Binary)**
Een binair bestandsformaat gebruikt door meteorologische diensten zoals het KNMI voor het distribueren van weermodellen en satellietdata. GRIB-bestanden vereisen specifieke parsers en worden vaak verwerkt via gespecialiseerde backend-systemen zoals in Python.

14 **Tile Serving**
Het serveren van kaartbeelden of 'tiles' in kleine stukjes op basis van zoomniveau en locatie. Belangrijk voor interactieve kaarten die performant moeten blijven bij grote datasets.

15 **Hot Reloading**
Een development feature waarbij codewijzigingen onmiddellijk worden toegepast zonder dat de volledige applicatie opnieuw hoeft te laden. Verhoogt ontwikkelsnelheid en verkort feedbackloops.

16 **Emerging Frameworks**
Nieuwe of snel evoluerende softwareframeworks die innovatieve benaderingen bieden voor webontwikkeling. Voorbeelden zijn Svelte, SolidJS en Qwik. Deze frameworks worden vaak vergeleken met gevestigde namen zoals React of Vue.js.


NHL
STENDEN
hogeschool