

Jonas Quintiens - Stage Ordolio

Realisatiedocument

Jonas Quintiens
Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling

Inhoudsopgave

1. INLEIDING	3
2. ANALYSE	4
2.1. Aanpak penetratietest	4
2.1.1. Scopebepaling	4
2.1.2. Methodologie	4
2.2. Analyse Wachtlijsten	5
2.2.1. Use Case Analyse van de wachtlijstfunctionaliteit	5
2.2.2. Business Case voor de wachtlijstfunctionaliteit	6
2.2.3. Onderzoek naar wachtlijst methodes	8
3. REALISATIE	9
3.1. Wachtlijst Functionaliteit voor evenementen	9
3.1.1. Technologische uitdagingen	18
3.1.2. Toekomstige uitbreidingen	19
3.2. Realisatie van de Pentest	19
3.2.1. Doel en context	19
3.2.2. Uitvoering	20
3.2.3. Resultaten en aanbevelingen	24
3.3. Overige Realisaties	24
3.3.1. Workflow Scrum	24
3.3.2. Uitgelichte opgeloste tickets	27
4. BESLUIT	32
LITERATUURLIJST	33
BIJLAGEN	34
4.1. Bijlage A – Security Fixes	35

1. Inleiding

Dit document beschrijft de concrete uitvoering van mijn stageproject bij Ordolio, een Belgische start-up die zich specialiseert in het digitaal ondersteunen van verenigingen. Het platform biedt oplossingen voor ledenbeheer, ticketverkoop, communicatie en evenementorganisatie. Ordolio richt zich voornamelijk op sportclubs, jeugdverenigingen en culturele organisaties, en streeft naar gebruiksvriendelijke tools die het administratieve werk van vrijwilligers en beheerders verlichten.

Mijn stage liep van 31 januari tot 30 mei 2025 en bood me de kans om actief mee te werken binnen het ontwikkelteam van Ordolio. Tijdens deze periode werkte ik aan twee kernopdrachten:

1. Het uitvoeren van een penetratietest op applicatieniveau om de beveiliging van het platform te evalueren.
2. Het ontwerpen en implementeren van een wachtlijstfunctionaliteit voor evenementen met beperkte capaciteit.

Naast deze hoofdpoddrachten droeg ik ook bij aan de dagelijkse werking van het ontwikkelteam. Ik loste bugs op, implementeerde kleinere features en ondersteunde collega's bij technische vraagstukken. Deze bijdragen gaven me de kans om vertrouwd te raken met de volledige ontwikkelcyclus binnen een professionele context.

Deze realisatie bouwt voort op het eerder opgestelde projectplan, waarin de context en doelstellingen reeds zijn toegelicht. In dit document ligt de nadruk op de technische uitvoering: welke keuzes werden gemaakt, hoe de implementatie verliep, welke tools en methodes werden ingezet, en welke resultaten dit opleverde.

De structuur van dit document is als volgt:

- In hoofdstuk 2 (Analyse) bespreek ik de gebruikte methodologieën en tools, waaronder de Weighted Ranking Methode (WRM) voor het bepalen van de meest geschikte wachtlijststrategie.
- In hoofdstuk 3 (Realisatie) licht ik de technische implementatie toe, geïllustreerd met voorbeelden en screenshots.
- In hoofdstuk 4 (Besluit) reflecteer ik op de impact van mijn werk, de behaalde resultaten en formuleer ik aanbevelingen voor de toekomst.

2. Analyse

Om de technische realisatie goed te kunnen begrijpen, is het belangrijk eerst stil te staan bij de gekozen aanpak en de onderliggende analyse. In het volgende hoofdstuk bespreek ik daarom de methodologieën en tools die ik gebruikte om tot een onderbouwde implementatie te komen.

2.1. Aanpak penetratietest

Om de veiligheid van de Ordolio-webapplicatie te evalueren, heb ik een penetratietest voorbereid volgens een gestructureerde en onderbouwde aanpak. Deze analyse beschrijft de gekozen methodologie, gebruikte tools en de afbakening van de testomgeving.

De test werd opgezet op basis van de richtlijnen van de **OWASP Top 10** en **NIST SP 800-115**, en combineerde zowel geautomatiseerde als handmatige technieken. De focus lag op het identificeren van kwetsbaarheden in de webapplicatie, API's en mobiele componenten, zonder impact op de productieomgeving.

2.1.1. Scopebepaling

Om de test beheersbaar en relevant te houden, heb ik vooraf een duidelijke scope afgebakend. De volgende onderdelen werden expliciet opgenomen in de test:

- **Webapplicatie:** inclusief authenticatie, gebruikersrollen, sessiebeheer en client-side beveiliging.
- **API-endpoints:** met focus op toegangscontrole, inputvalidatie en datalekken.
- **Mobiele applicatie:** communicatie met backend, lokale opslag en reverse engineering.

Buiten scope vielen onder andere infrastructuurbeveiliging (zoals firewalls en servers), interne netwerken, externe SaaS-diensten (zoals Stripe), en fysieke of sociale aanvallen (zoals phishing of tailgating).

2.1.2. Methodologie

De testaanpak bestond uit drie fasen:

1. Verkenning en informatieverzameling

In deze fase werd het aanvalsoppervlak in kaart gebracht. Dit omvatte:

- Extractie van alle URL-endpoints via een eigen script.
- Analyse van publieke informatie (OSINT).
- Identificatie van API-routes, loginpagina's en gebruikersrollen.

2. Geautomatiseerde kwetsbaarheidsscans

Diverse tools werden ingezet om snel bekende kwetsbaarheden te detecteren:

- **OWASP ZAP** en **Burp Suite** voor dynamische applicatiescans.
- **Snyk** voor statische code-analyse van afhankelijkheden.
- **Security Header Scan** en **SSL Labs** voor configuratiecontroles.

3. Handmatige validatie en logische tests

Omdat geautomatiseerde tools niet alle contextspecifieke fouten detecteren, werd aanvullend getest op:

- Onjuiste toegangscontrole (bv. roloverschrijding of IDOR).
- Invoervalidatieproblemen zoals XSS of SQL injection.
- Business logic flaws, zoals onterecht toegestane acties of privilege escalation.

Deze aanpak maakte het mogelijk om niet enkel technische kwetsbaarheden te detecteren, maar ook logische fouten die enkel zichtbaar worden in de context van de applicatie. De resultaten van deze analyse vormden de basis voor de uitvoering van de penetratietest, die verder wordt toegelicht in het realisatiehoofdstuk.

2.2. Analyse Wachtlijsten

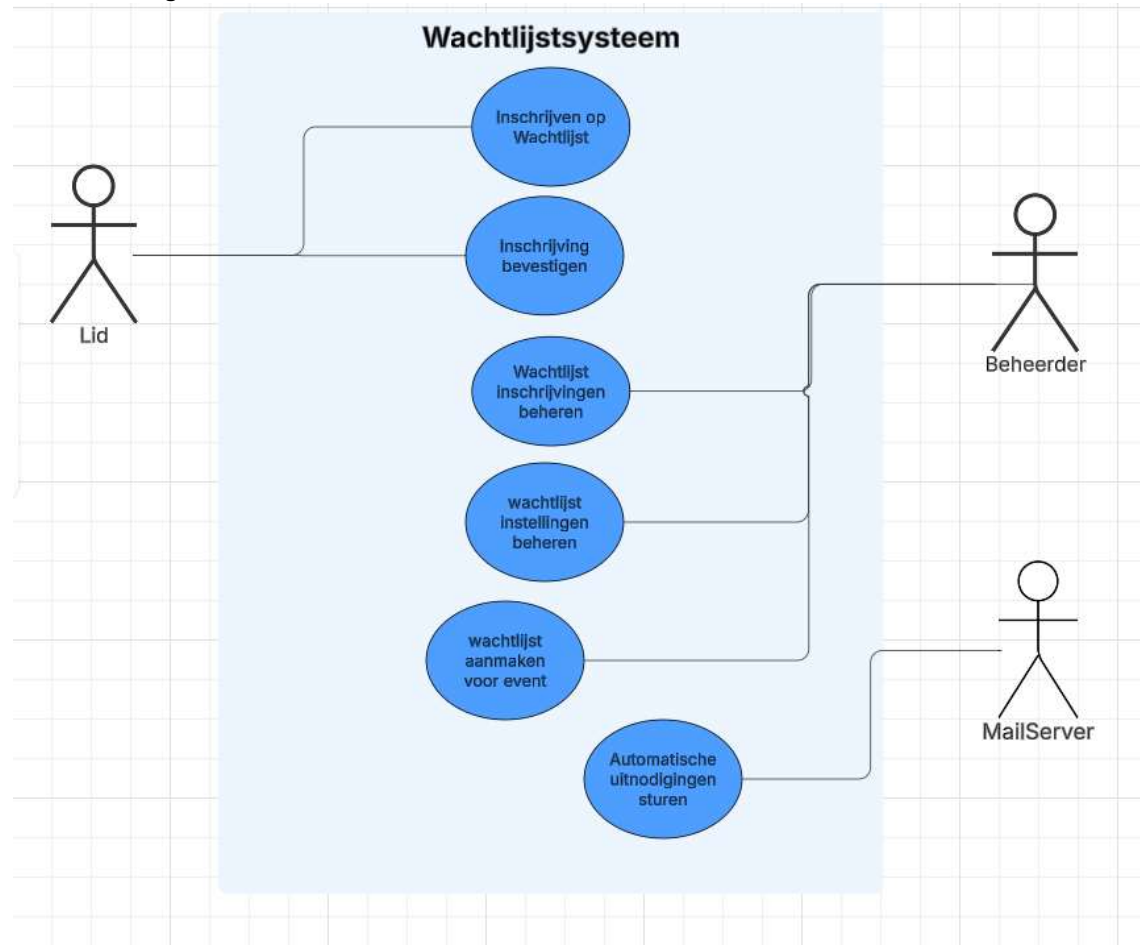
2.2.1. Use Case Analyse van de wachtlijstfunctionaliteit

Om de vereisten en verantwoordelijkheden van het wachtlijststelsel grondig in kaart te brengen, ontwikkelde ik een use case diagram. Dit diagram toont de interacties tussen de verschillende actoren en het systeem, en helpt bij het structureren van de functionele vereisten nog vóór de technische implementatie. Het model werd opgesteld op basis van gesprekken met de stakeholders binnen Ordolio en de bestaande platformstructuur.

Gedefinieerde actoren

- **Gebruiker (Lid of Bezoeker)**
 - Kan zich inschrijven op de wachtlijst bij een volzet evenement.
 - Ontvangt eventueel een uitnodiging per e-mail om alsnog een ticket te reserveren.
 - Kan via een tokenlink tijdelijk een ticket aankopen.
- **Beheerder (Eventorganisator)**
 - Activeert de wachtlijst per evenement.
 - Kan capaciteit aanpassen en eventueel handmatig gebruikers uitnodigen.
 - Raadpleegt de status van de wachtlijst via het beheerpaneel.
- **Systeem (Ordolio Platform)**
 - Detecteert automatisch wanneer de capaciteit van een event is bereikt.
 - Registreert inschrijvingen op de wachtlijst.
 - Verstuurde geautomatiseerde uitnodigingen op basis van het FIFO-principe.
 - Beheert tokens en bewaakt de geldigheidsduur van uitnodigingen.

Use Case Diagram



Nut van de analyse

De use case analyse hielp niet alleen bij het structureren van de initiële MVP, maar diende ook als basis voor latere uitbreidingen, zoals het overwegen van alternatieve methodes (batch invites, prioriteitslijsten). Dankzij de duidelijke actoren en verantwoordelijkheden konden logica en verantwoordelijkheidsverdeling in de code efficiënter uitgewerkt worden.

2.2.2. Business Case voor de wachtlijstfunctionaliteit

Naast de technische analyse is het belangrijk om stil te staan bij de strategische en economische meerwaarde van de wachtlijstfunctionaliteit. Deze business case onderzoekt waarom het toevoegen van een wachtlijst niet alleen zinvol is vanuit gebruikersstandpunt, maar ook een concreet voordeel oplevert voor de organisatie.

Doel van de functionaliteit

De wachtlijstfunctionaliteit werd ontwikkeld om beter om te gaan met de beperkte capaciteit van populaire evenementen. In de oorspronkelijke situatie gingen vrijkomende plaatsen vaak verloren, omdat er geen gestructureerde opvolging was. Dit leidde tot gemiste inkomsten, ontevreden gebruikers en verhoogde werklast voor eventbeheerders.

Voordelen voor de organisatie

- 1. Hogere bezettingsgraad van evenementen**
Door automatisch deelnemers uit te nodigen bij vrijkomende plaatsen, wordt het risico op lege stoelen of ongebruikte tickets drastisch verminderd. Dit maximaliseert de capaciteit zonder manuele opvolging.
- 2. Betere gebruikerservaring**
Gebruikers hoeven niet constant de beschikbaarheid te controleren. Ze worden automatisch verwittigd wanneer er een plaats vrijkomt. Dit verhoogt de tevredenheid en het vertrouwen in het platform.
- 3. Efficiëntere werking voor beheerders**
Eventorganisatoren hoeven niet handmatig deelnemers op te volgen of mails te sturen. Het systeem automatiseert de volledige flow, van inschrijving tot uitnodiging. Dit bespaart tijd en vermindert fouten.
- 4. Data-inzichten en optimalisatie**
Door het gebruik van de wachtlijst ontstaat er waardevolle data over de vraag naar specifieke evenementen. Dit laat toe om beter te plannen, nieuwe sessies toe te voegen of inschrijvingsbeleid aan te passen op basis van reële interesse.
- 5. Concurrentieel voordeel**
Een slimme wachtlijstoplossing positioneert Ordolio sterker tegenover andere aanbieders van eventbeheer. Veel platformen bieden enkel een simpele “volzet”-melding, zonder opvolging. Dit systeem maakt het verschil door efficiëntie en klantgerichtheid te combineren.

Kosten-batenanalyse

Aspect	Kosten	Baten
Ontwikkeling	Initiële ontwikkeltijd en integratie in bestaande code	Automatische opvolging, minder manuele tussenkomst
Onderhoud	Beperkt: enkele validaties, tokens en e-mails	Robuuste werking, weinig foutgevoeligheid
Gebruikersimpact	Nieuwe interface-elementen en communicatieflow	Hoger gebruiksgemak en inschrijvingszekerheid
Commercieel	Geen directe kost, wel extra dev-inspanning	Meer inschrijvingen, verhoogde conversie en retentie

De business case toont aan dat de wachtlijstfunctionaliteit niet alleen een technisch project is, maar ook duidelijke voordelen biedt op vlak van gebruikservaring, organisatorische efficiëntie en commerciële waarde. De implementatie van een FIFO-gebaseerde MVP zorgt voor snelle impact, terwijl het systeem tegelijk ruimte biedt voor toekomstige uitbreidingen (zoals prioriteit of batch invites) op basis van organisatorische noden.

2.2.3. Onderzoek naar wachtlijst methodes

Bij het ontwikkelen van een wachtlijstfunctionaliteit binnen een evenementenplatform is het essentieel om een methode te kiezen die zowel eerlijk als efficiënt is. De manier waarop gebruikers op een wachtlijst worden geplaatst en uitgenodigd, heeft een directe impact op de gebruikerservaring, de beheersbaarheid van het systeem en de technische haalbaarheid van de implementatie.

Om tot een onderbouwde keuze te komen, voerde ik een vergelijkend onderzoek uit naar vier gangbare benaderingen. Elke methode werd beoordeeld op basis van zes zorgvuldig gekozen criteria, waaronder eerlijkheid, technische complexiteit en het risico op lege plaatsen. Deze criteria kregen een gewogen waarde binnen een **Weighted Ranking Methode (WRM)**, zodat de uiteindelijke keuze niet alleen subjectief, maar ook kwantitatief onderbouwd is.

De vier onderzochte methodes zijn:

1. **FIFO (First In, First Out)**
Gebruikers worden in volgorde van inschrijving op de wachtlijst geplaatst. Wie zich het eerst aanmeldt, krijgt als eerste de kans om in te schrijven zodra er een plaats vrijkomt. Deze methode is eenvoudig, transparant en technisch makkelijk te implementeren.
2. **Prioriteitswachtlijst**
Gebruikers krijgen een score of label (bijvoorbeeld VIP, trouwe klant), en worden op basis daarvan hoger in de lijst geplaatst. Dit systeem kan eerlijk aanvoelen voor specifieke doelgroepen, maar vereist een complexere configuratie en beheerlogica.
3. **Loting**
Bij het vrijkomen van een plaats wordt willekeurig iemand uit de wachtlijst gekozen. Dit garandeert gelijke kansen, maar kan als oneerlijk worden ervaren door gebruikers die zich vroeg inschreven.
4. **Batch invites / Golven**
Gebruikers worden in groepen (bijvoorbeeld per 5) uitgenodigd om zich in te schrijven. Dit verhoogt de kans op snelle invulling van plaatsen, maar vereist een goed afgestemde communicatieflow.

VERGELIJKINGSCRITERIA

Elke methode werd beoordeeld op zes criteria, met een gewicht toegekend via de **Weighted Ranking Methode (WRM)**:

Criterium	Gewicht	FIFO	Prioriteit	Loting	Batch/Golf
Eerlijkheid	25%	2.5	0.25	1.75	2.0
Beheercomplexiteit	20%	2.0	1.0	1.8	1.4
Gebruikerservaring	15%	1.2	0.9	0.9	1.35
Technische complexiteit	15%	1.5	0.15	0.75	0.75
Flexibiliteit	15%	0.6	1.5	0.15	0.75
Risico op lege plaatsen	10%	0.8	0.7	0.4	1.0
Totaalscore		8.6	4.5	5.75	7.25

ANALYSE EN CONCLUSIE

De analyse toont duidelijk aan dat de **FIFO-methode** de hoogste totaalscore behaalt (8.6/10). De belangrijkste voordelen zijn:

- **Eenvoudige implementatie:** De logica is rechtlijnig en vereist weinig extra configuratie.
- **Transparantie:** Gebruikers begrijpen intuïtief hoe de volgorde werkt.
- **Lage technische en beheercomplexiteit:** Zowel voor ontwikkelaars als beheerders is deze methode makkelijk te hanteren.
- **Schaalbaarheid:** FIFO kan eenvoudig worden uitgebreid met extra logica, zoals prioriteitsniveaus of automatische opvolging.

Hoewel alternatieven zoals **Batch invites** ook goed scoren op bepaalde vlakken (zoals het vermijden van lege plaatsen), blijft FIFO de meest robuuste en flexibele oplossing voor de context van Ordolio, waar eenvoud, snelheid en duidelijkheid centraal staan.

TECHNOLOGISCHE KEUZES

Voor zowel de pentest als de implementatie van de nieuwe functionaliteit maakte ik gebruik van technologieën die passen binnen de bestaande stack van Ordolio:

Domein	Technologie/tool
Backend	Django, Django REST Framework
Frontend	Flutter, Django
Beveiliging	OWASP ZAP, Burp Suite, Snyk
Containerisatie	Docker
Workflow en versiebeheer	Jira, Git

3. Realisatie

3.1. Wachtlijst Functionaliteit voor evenementen

De wachtlijstfunctionaliteit werd ontwikkeld als een **Minimum Viable Product (MVP)**, gericht op het vereenvoudigen van het beheer van volzette evenementen. De implementatie werd opgebouwd binnen de bestaande infrastructuur van de Ordolio-webapplicatie, met Django als backend en gebruik van Django-templates voor de frontend.

ACTIVATIE EN CONFIGURATIE

Bij het aanmaken van een nieuw evenement krijgt de beheerder de optie om een wachtlijst te activeren. Daarbij kan een maximumcapaciteit worden ingesteld. Zowel deze activering als de limiet kunnen nadien nog aangepast worden via de eventinstellingen. In onderstaande figuren zie je de schermen die hier van toepassing zijn.

Dashboard
Kalender
Nieuws
Admin
Begeleider

Airssoft Bierbeek

FAQ
Halo, Airsoft Bierbeek
A

Voorpagina

Ledenbeheer

Mailing

Evenementen & Activiteiten

Werkingsactiviteiten

Nieuws

Foto's

Kalender

Partner deals

Financiën

Facturatie

Verbruik

Instellingen

Gebruikers

Verkoop start op

17/05/2025

0:00

Verkoop eindigt op

18/05/2025

23:00

Maximaal aantal deelnemers

100

Tickets
Geef aan welke tickets er zijn voor dit evenement.

+ Voeg een ticket toe

Ticket	Prijs	Aantal beschikbaar	Enkel voor leden?	Acties
Voeg tickets toe door op de knop rechtsboven te klikken.				

Wachttijst
Kunnen mensen zich op een wachtlijst inschrijven als het uitverkocht is?

Je hebt de mogelijkheid om een wachtlijst te gebruiken voor dit evenement. Als het evenement uitverkocht is, kunnen mensen zich op de wachtlijst inschrijven. Je kan deze mensen dan later contacteren als er plaatsen vrijkomen.

☒ Gebruik een wachtlijst schakel een wachtlijst in wanneer het evenement is uitverkocht.

Maximum aantal wachtlijst inschrijvingen

10

Laat leeg voor onbeperkt. Dit is het totale aantal mensen dat op de wachtlijst kan staan.

Fiscale- en deelnameattesten
Moeten deelnemers attesten ontvangen?

Genereer fiscale- en deelnameattesten

Kies dit als:

Je een activiteit organiseert
 Je een kamp organiseert

Genereer GEEN attesten

Kies dit als:

Je een evenement organiseert waarbij geen attesten nodig zijn
 Je tickets verkoopt

Figuur 1: Activeer de wachtlijst bij het aanmaken van een event.

Algemeen

Betalingen

Thema

Emails

Wachtlijst

Wachtlijst instellingen

Kunnen mensen zich op een wachtlijst inschrijven als het evenement volzet is?

Opslaan

Met een wachtlijst kunnen mensen zich inschrijven wanneer het evenement volzet is. Wanneer er plaatsen vrijkomen, kan je deze mensen verwittigen.



☒ Gebruik een wachtlijst Schakel een wachtlijst in wanneer het evenement volzet is

Maximum aantal wachtlijst inschrijvingen

10

Laat leeg voor onbeperkt. Dit is het totale aantal mensen dat op de wachtlijst kan staan.

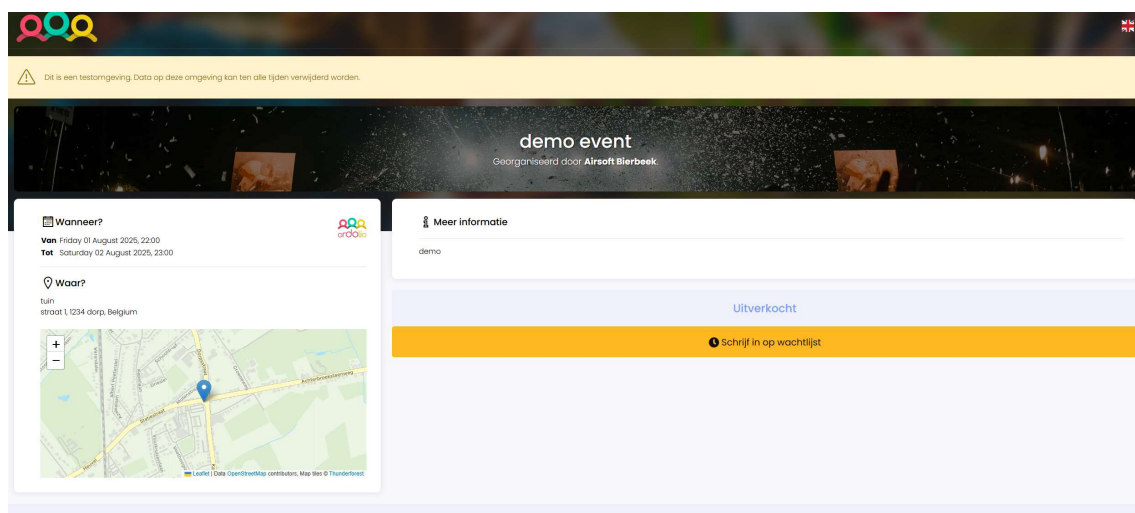
Wachtlijst inschrijvingen

Naam	E-mail	Status	Datum	Acties
jonas	jonas@ordollo.com	Wachtend	17/05/2025 18:36	 

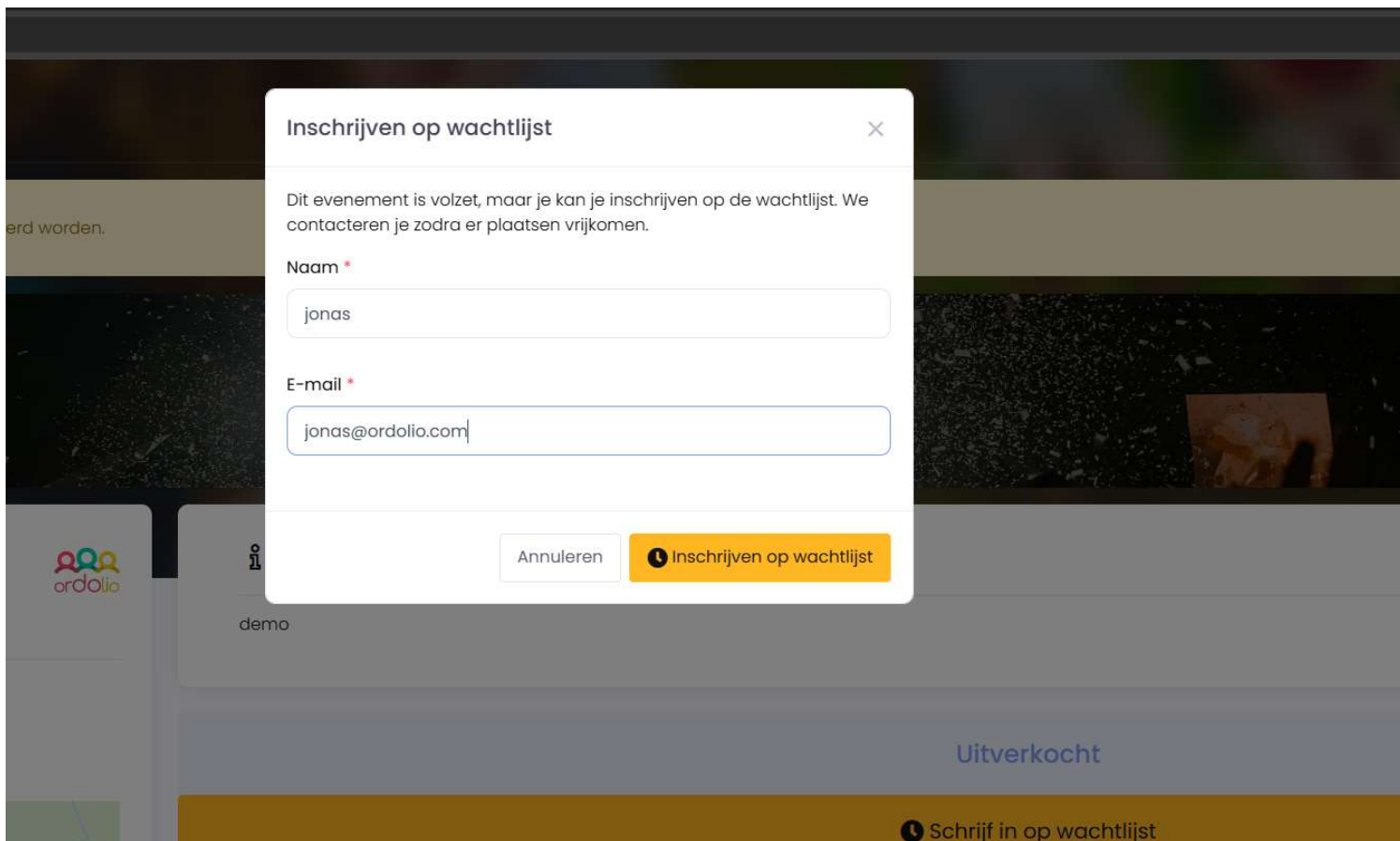
Figuur 2: Activeer de wachtlijst retroactief

INSCHRIJVING OP DE WACHTLIJST

Zodra de capaciteit van een evenement bereikt is, wordt de bestelpagina automatisch aangepast: gebruikers krijgen dan de mogelijkheid om zich op de wachtlijst in te schrijven. Voor ingelogde gebruikers worden relevante gegevens vooraf ingevuld. Na bevestiging wordt hun inschrijving geregistreerd in een nieuw aangemaakt Wachtlijst-model, dat gelinkt is aan het betreffende evenement. Hieronder enkele screenshots van de functionaliteiten.



Figuur 3: Publieke pagina voor een volzet evenement met een geactiveerde wachtlijst.



Figuur 4: Modal voor inschrijving op de wachtlijst.

Bevestiging wachtlijst voor demo event Σ Inbox x



Airsoft Bierbeek

aan mij ▾

Bevestiging wachtlijst

Beste tester,

Je bent succesvol toegevoegd aan de wachtlijst voor "demo event".

We contacteren je zodra er plaatsen vrijkomen. Je hoeft verder niets te doen.

Met vriendelijke groeten,
Airsoft Bierbeek



Virus-free www.avast.com

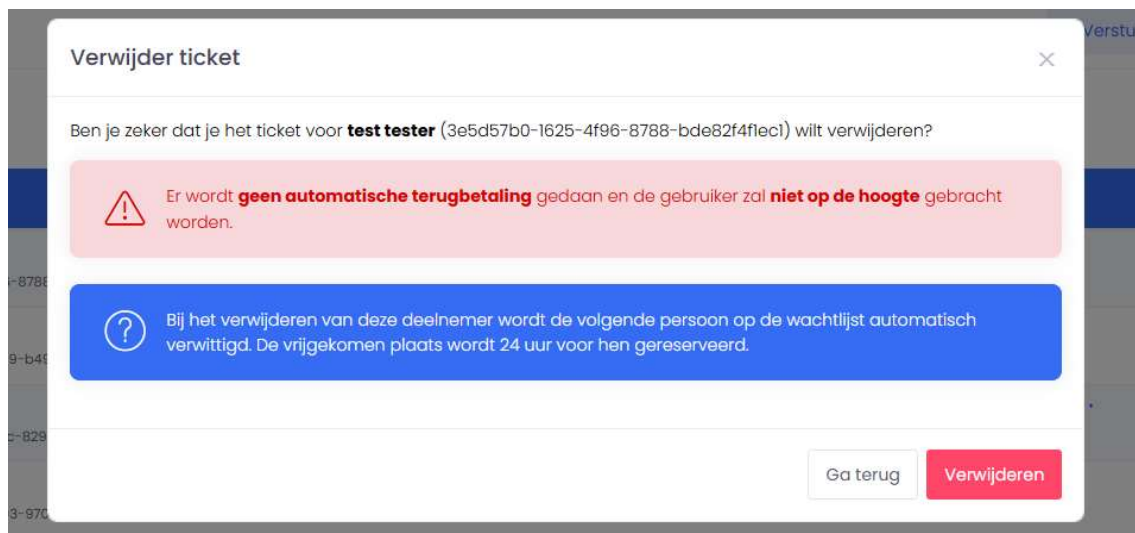
Figuur 5: bevestigingsmail na inschrijving.

BEHEER VAN VRIJGEKOMEN PLAATSEN

Wanneer een gebruiker zijn inschrijving annuleert of een ticket wordt terugbetaald, controleert het systeem of er een actieve wachtlijst bestaat. Indien dat het geval is, wordt op basis van het FIFO-principe de eerstvolgende persoon uit de wachtlijst geselecteerd.

Deze gebruiker ontvangt een automatische e-mail met een unieke link naar de bestelpagina. De link bevat een tijdelijke toegangstoken, waarmee de gebruiker tijdelijk het recht krijgt om een of meerdere tickets aan te kopen. Gedurende deze reservatieperiode is de plaats geblokkeerd voor andere gebruikers.

Indien de uitnodiging verloopt zonder dat er een bestelling gebeurt, wordt de volgende persoon uit de lijst aangeschreven. Als de wachtlijst leeg is, wordt het evenement opnieuw opengesteld voor reguliere inschrijving.



Figuur 6: (in de blauwe kader) melding aan de beheerder bij het verwijderen van een inschrijving.

Tickets beschikbaar voor demo event 🔔 Inbox x



Airsoft Bierbeek

aan mij ▼

Tickets beschikbaar!

Beste tester,

Er zijn tickets beschikbaar gekomen voor "demo event". Je staat op de wachtlijst voor dit evenement.

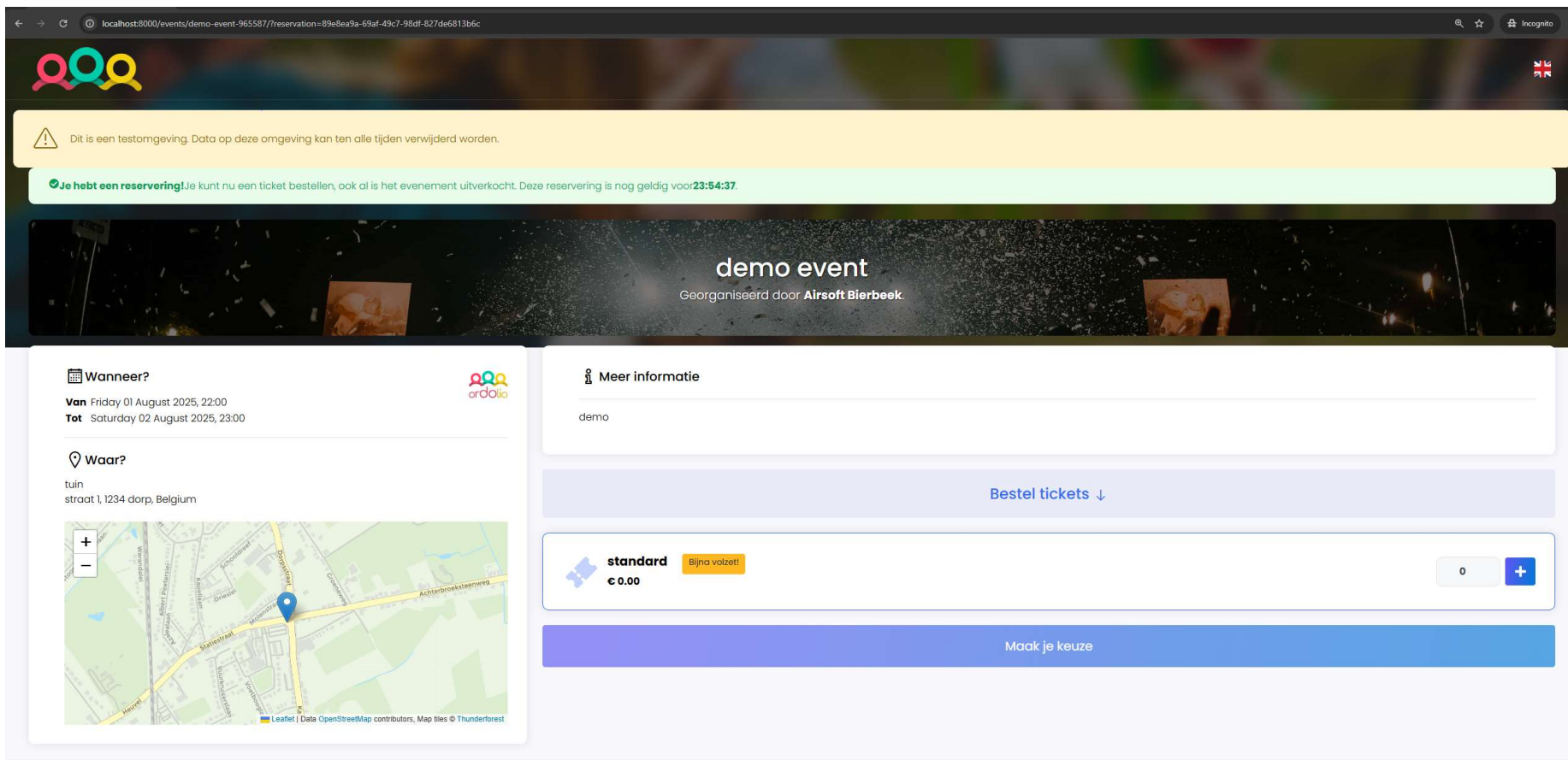
Klik op onderstaande knop om naar de ticketpagina te gaan en je ticket(s) te bestellen:

[Naar ticketpagina](#)

Let op: De beschikbare tickets kunnen snel uitverkocht zijn. We kunnen niet garanderen dat er nog tickets beschikbaar zijn als je deze mail later leest.

Met vriendelijke groeten,
Airsoft Bierbeek

Figuur 7: Uitnodigingsmail met magic link om tickets te bestellen.



Figuur 8: Publieke pagina met een geldige reservatie token

TECHNISCHE IMPLEMENTATIE (KERNPUNTEN)

- **Modeluitbreiding:** Een nieuw Wachtlijst-model werd gecreëerd in Django, met een ForeignKey-relatie naar het Event-model.
- **Frontend:** De inschrijvingsflow werd toegevoegd aan de bestaande Django-templates, met meldingen voor volzette evenementen en bevestiging van inschrijving op de wachtlijst.
- **Tokenbeheer:** Tokens worden automatisch gegenereerd bij het versturen van uitnodigingen, met geldigheid op basis van een tijdslimiet.
- **Mailverkeer:** Uitnodigingen worden verstuurd via de bestaande e-mailsystemen, met een gepersonaliseerde link voor directe toegang tot de reservering.
- **Beheerscherm:** Voor organisatoren werd een overzicht toegevoegd waarin de huidige wachtlijststatus per event zichtbaar is, met optionele manuele acties.

Deze MVP legt een robuuste basis voor verdere uitbreidingen, zoals ondersteuning van meerdere types wachtlijsten of automatische opvolging via batchverwerking.

3.1.1. Technologische uitdagingen

De implementatie van de wachtlijstfunctionaliteit bracht enkele onverwachte technische uitdagingen met zich mee, voornamelijk als gevolg van de bestaande architectuur van het platform.

De Ordolio-webapplicatie bevat reeds een uitgebreide set aan functionaliteiten rond ticketverkoop, toegangscontrole, ledenbeheer en evenementinstellingen. Veel van deze componenten beïnvloeden direct of indirect de logica rond het bestellen van tickets en het bepalen of een evenement al dan niet is uitverkocht.

Deze verwevenheid zorgde ervoor dat het toevoegen van een nieuwe laag, de wachtlijst, leidde tot onverwachte bugs en edge cases. Zo bleken bepaalde validaties of automatische acties (zoals het blokkeren van bestellingen bij volzette events) niet compatibel met de nieuwe logica rond reserveringen via de wachtlijst.

Ook het correct updaten van de status van een evenement na het vrijvallen van een plaats vereiste aanpassingen op meerdere plaatsen in de codebase.

Om deze problemen op te lossen, was het nodig om diepgaand inzicht te krijgen in de bestaande logica en afhankelijkheden. Dit resulteerde in een reeks refactorings en extra validaties, zodat de nieuwe functionaliteit robuust kon worden geïntegreerd zonder bestaande processen te verstoren.

3.1.2. Toekomstige uitbreidingen

Hoewel de huidige implementatie gebaseerd is op het FIFO-principe, laat de onderliggende architectuur ruimte voor verdere uitbreidingen. Op basis van het eerder uitgevoerde vergelijkend onderzoek zijn er verschillende alternatieve methodes die in de toekomst kunnen worden toegevoegd, afhankelijk van de noden van de organisatie:

- **Prioriteitswachlijsten**
Gebruikers kunnen een score of label krijgen (bijvoorbeeld “VIP” of “trouwe deelnemer”), waardoor ze hoger in de wachtlijst terechtkomen. Dit vereist een uitbreiding van het datamodel en een aangepaste selectieprocedure bij het vrijvallen van plaatsen.
- **Loting**
In plaats van een vaste volgorde, wordt bij het vrijkomen van een plaats willekeurig iemand uit de wachtlijst gekozen. Dit kan eerlijker aanvoelen in situaties met grote vraag, maar vereist een robuuste randomisatie en logging.
- **Batch invites (golven)**
Gebruikers worden in groepen uitgenodigd om zich in te schrijven. Dit verhoogt de kans op snelle invulling van plaatsen en kan gecombineerd worden met een tijdslimiet per groep.
- **Automatische opvolging en reminders**
Indien een gebruiker niet tijdig reageert op een uitnodiging, kan het systeem automatisch de volgende persoon uitnodigen of een herinnering sturen. Dit verhoogt de efficiëntie en vermindert het risico op lege plaatsen.

Dankzij de modulaire opbouw van de huidige MVP is het technisch mogelijk om deze uitbreidingen gefaseerd te implementeren, zonder de bestaande werking te verstoren. Dit maakt het systeem toekomstbestendig en aanpasbaar aan veranderende behoeften.

3.2. Realisatie van de Pentest

Tijdens mijn stage bij Ordolio ontdekte ik toevallig een stored XSS-kwetsbaarheid tijdens reguliere ontwikkeltesten. Deze kwetsbaarheid gaf directe toegang tot de DOM zonder aanvullende exploitatie. Hoewel dit probleem snel werd opgelost, vormde het de aanleiding voor een uitgebreide penetratietest van het platform. Het doel: een diepgaand inzicht krijgen in de beveiligingsstatus van de webapplicatie, kwetsbaarheden opsporen, en aanbevelingen formuleren om de dataveiligheid structureel te versterken.

De initiële kwetsbaarheid werd opgelost door het aanpassen van de invoervalidatie en het ontsnappen van gebruikersinput. De exacte wijzigingen zijn zichtbaar in de codeverschillen die hieronder als screenshots zijn opgenomen (zie **Figuur 1** en **Figuur 2** in [Bijlage A – Security Fixes](#)). Deze tonen de aangepaste backendvalidatie.

3.2.1. Doel en context

Ordolio verwerkt persoonsgegevens van verenigingsleden, waaronder medische info, contactgegevens en betalingsdata. De webapplicatie is opgebouwd met **Django** en gebruikt **PostgreSQL** via de Django ORM. Daarnaast is er een **Expo-gebaseerde mobiele applicatie** die communiceert met een REST API (Django REST Framework). De hele omgeving draait in **Docker containers** via docker-compose, wat ook netwerk- en containerbeveiliging relevant maakt.

3.2.2. Uitvoering

De test werd uitgevoerd in een veilige **stagingomgeving**. De pentest richtte zich op de volledige applicatiearchitectuur, inclusief frontend, backend, API, containerconfiguratie en e-mailbeveiliging.

ATTACK SURFACE MAPPING

Het aanvalsoppervlak werd in kaart gebracht aan de hand van:

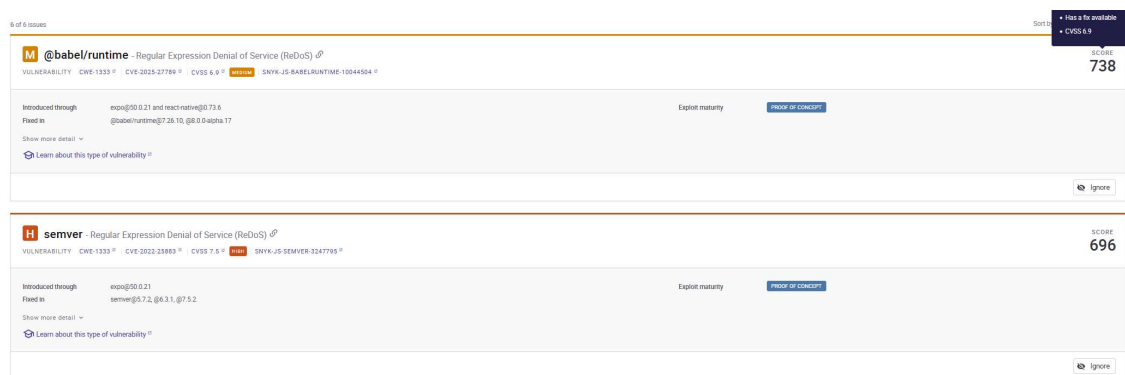
- Webroutes en endpoints (1393 URL's automatisch geëxtraheerd via custom script)
- API-endpoints voor mobiele communicatie
- Invoervelden: formulieren, zoekvelden, headers, cookies
- Docker-netwerkstructuur en container-exposure
- Externe integraties zoals **Stripe**

STATISCHE ANALYSE

Via **Snyk Code** werd de codebase gescand op:

- SQL-injectie
- Cross-Site Scripting (XSS)
- Hardcoded secrets
- Insecure deserialization

Snyk gaf contextgevoelige aanbevelingen op basis van ernst en exploitability. Onder andere een ReDos en een minder ernstige Cross-site Scripting (reflected ipv stored) kwetsbaarheid.



Figuur 9: Snyk code resultaten.

AUTOMATISCHE EN HANDMATIGE SCANS

Er werden diverse tools gebruikt:

- **OWASP ZAP**: baseline kwetsbaarheidsscan
- **Burp Suite (CE)**: handmatige manipulatie en fuzzing
- **Security Header Scan**: controle op veilige headers
- **SSL Labs Scan**: evaluatie van HTTPS-configuratie

E-MAILBEVEILIGING

E-mail is een veelvoorkomend aanvalsvector voor cybercriminelen. Een zwakke configuratie kan leiden tot phishing, spoofing of zelfs het misbruiken van een onbeveiligde e-mailserver als relay voor spam en kwaadaardige berichten. Daarom worden SPF, DKIM en DMARC gebruikt om te verifiëren dat verzonden e-mails afkomstig zijn van legitieme bronnen en niet kunnen worden vervalst.

De e-mailserver werd beoordeeld op:

- **SPF**: correct geconfigureerd, maar verwijzing naar potentieel kwetsbare IP-range
- **DKIM**: ontbreekt
- **DMARC**: ingesteld op "none", dus geen bescherming tegen spoofing
- **SMTP relay**: correct gesloten voor externe misbruik

AANVULLENDE CUSTOM TESTEN

Naast het gebruik van standaardtools zoals OWASP ZAP, Burp Suite en Snyk, ontwikkelde ik ook enkele **custom scripts en tools** om de pentest te verdiepen en specifieke aspecten van de applicatie te analyseren:

- **URL-extractiescript (Python)**
Om een volledig overzicht te krijgen van het aanvalsoppervlak, schreef ik een eigen Python-script dat automatisch alle URL's in de applicatie verzamelde. Dit script doorzocht de HTML-responses en JavaScript-bestanden op links en routes, en leverde uiteindelijk een lijst op van **1 393 unieke endpoints**. Deze lijst werd gebruikt als input voor verdere geautomatiseerde en handmatige tests.
- **Security Header Checker (webtool)**
Voor het controleren van HTTP-responsheaders ontwikkelde ik een eigen webpagina die de aanwezigheid en configuratie van belangrijke beveiligingsheaders controleert.

De volgende headers worden gecontroleerd:

- **X-Frame-Options** – voorkomt clickjacking via iframes
- **Content-Security-Policy** (met focus op frame-ancestors) – controleert welke domeinen content mogen insluiten
- **Strict-Transport-Security** – dwingt HTTPS af
- **X-Content-Type-Options** – voorkomt MIME-sniffing
- **Referrer-Policy** – bepaalt welke referrer-informatie wordt meegestuurd
- **Permissions-Policy** – beperkt toegang tot browser-API's

De tool bevat ook een **iframe loading test**, waarmee gecontroleerd wordt of de applicatie in een iframe geladen kan worden, wat een indicatie kan zijn van ontbrekende of incorrect ingestelde headers. Deze custom tools boden extra inzichten die niet altijd door standaardscanners worden opgemerkt, en droegen bij aan een vollediger beeld van de beveiligingsstatus van de applicatie.

Iframe Loading test

Security Headers to Check:

- **X-Frame-Options** - Controls whether a page can be displayed in iframes
- **Content-Security-Policy** - Specifically the frame-ancestors directive
- **Strict-Transport-Security** - Forces HTTPS connections
- **X-Content-Type-Options** - Prevents MIME type sniffing
- **Referrer-Policy** - Controls referrer information
- **Permissions-Policy** - Controls available features and APIs

These headers can be checked using browser developer tools (Network tab) or with this command:

```
curl -I https://example.com
```

Note: Due to browser CORS restrictions, we need to use a proxy or backend to check headers. Below

Check All Headers

Results will appear below each iframe

Production Environment

Figuur 10: web tool om security headers te testen.

```

get_urls.py
1  import os
2  import re
3
4  # Define the directory to search
5  directory = 'd:/files/ordolio/ordolio'
6
7  # Define the regex pattern to match URL patterns
8  url_pattern = re.compile(r'path\((([^\)]+)\))')
9
10 # Function to search for URL patterns in a file
11 def search_file(file_path):
12     with open(file_path, 'r', encoding='utf-8') as file:
13         content = file.read()
14         matches = url_pattern.findall(content)
15         return [(match, file_path) for match in matches]
16
17 # Function to search for URL patterns in a directory
18 def search_directory(directory):
19     url_patterns = []
20     for root, _, files in os.walk(directory):
21         for file in files:
22             if file.endswith('.py'):
23                 file_path = os.path.join(root, file)
24                 url_patterns.extend(search_file(file_path))
25     return url_patterns
26
27 # Extract URL patterns
28 url_patterns = search_directory(directory)
29
30 # Function to format the URL pattern
31 def format_url_pattern(pattern, file_path):
32     parts = pattern.split(',')
33     url_path = parts[0].strip().strip('"\'')
34     view = parts[1].strip() if len(parts) > 1 else 'N/A'
35     name = parts[2].strip() if len(parts) > 2 else 'N/A'
36     example_url = url_path
37     if '<' in url_path and '>' in url_path:
38         example_url = re.sub(r'<[>]+>', 'example', url_path)
39     relative_file_path = os.path.relpath(file_path, directory)
40     return f"URL: {url_path}\nView: {view}\nName: {name}\nExample: {example_url}\nFile: //{relative_file_path}\n"
41
42 # Print the formatted URL patterns
43 for pattern, file_path in url_patterns:
44     try:
45         print(format_url_pattern(pattern, file_path))
46     except Exception as e:
47         print(f"Error processing pattern: {pattern} in file: {file_path}\nError: {e}\n")

```

Figuur 11: Python scrip om URLs te identificeren.

3.2.3. Resultaten en aanbevelingen

De test leverde de volgende bevindingen en aanbevelingen op:

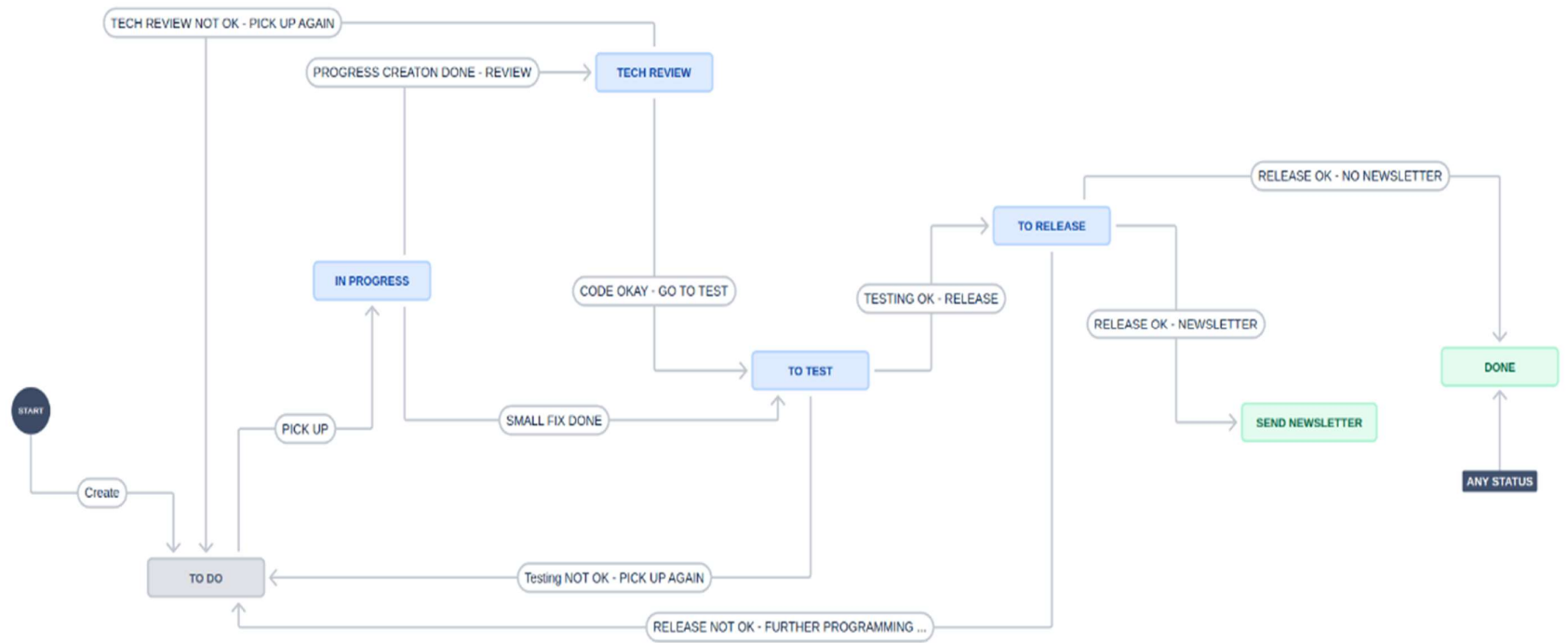
Onderdeel	Bevinding	Aanbeveling
Invoervalidatie	Onvoldoende streng op sommige endpoints, vermijd het gebruik van niet gesaniteerde user input in het renderen van html responses.	Uniforme server-side validatie op alle inputvelden
E-mailbeveiliging	DKIM ontbreekt, DMARC te zwak	DKIM configureren, DMARC op <i>quarantine</i> of <i>reject</i> zetten
Beveiligingsheaders	CSP ontbreekt op frontend	Implementeer Content Security Policy-header
Testbeleid	Geen vaste testcyclus aanwezig	Jaarlijkse penetratietest voorzien in ontwikkelbeleid

3.3. Overige Realisaties

Naast de twee hoofdopdrachten, de penetratietest en de ontwikkeling van de wachtlijstfunctionaliteit, heb ik tijdens mijn stageperiode ook actief bijgedragen aan de dagelijkse werking van het ontwikkelteam. Deze bijdragen omvatten het oplossen van kleinere bugs, het implementeren van nieuwe features en het ondersteunen van collega's bij technische vraagstukken. Hoewel deze taken op zichzelf minder omvangrijk zijn, vormden ze een belangrijk onderdeel van mijn leerproces en gaven ze me de kans om vertrouwd te raken met de volledige ontwikkelcyclus binnen een professionele context. In dit hoofdstuk licht ik enkele van deze realisaties toe.

3.3.1. Workflow Scrum

Tijdens mijn stage werkte ik binnen een team dat de Scrum-methodologie hanteerde als leidraad voor het ontwikkelproces. Deze agile werkwijze zorgde voor een gestructureerde en iteratieve aanpak, waarbij taken opgedeeld werden in kleinere werkpakketten en opgevolgd via het Jira-platform. Door actief deel te nemen aan deze workflow leerde ik niet alleen hoe ik mijn werk efficiënt kon plannen en documenteren, maar ook hoe belangrijk communicatie en samenwerking zijn binnen een ontwikkelteam. In dit hoofdstuk geef ik een overzicht van hoe de Scrum-workflow werd toegepast binnen Ordolio en hoe ik hieraan heb bijgedragen.



Figuur 12: QA flow via Jira

1. STARTFASE

- Het proces begint met de stap **START**, waarna een taak wordt gecreëerd (**Create**).
- De taak wordt vervolgens in de **TO DO**-status geplaatst, waar het klaarstaat om opgepakt te worden.

2. UITVOERINGSFASE

- Een taak wordt opgepakt (**PICK UP**) en gaat naar de status **IN PROGRESS**.
- Zodra de ontwikkeling klaar is, wordt een beoordeling uitgevoerd (**TECH REVIEW**).

3. TECHNISCHE REVIEW

- Als de technische review succesvol is (**CODE OKAY - GO TO TEST**), gaat de taak naar de testfase (**TO TEST**).
- Als de code niet goedgekeurd wordt (**TECH REVIEW NOT OK - PICK UP AGAIN**), wordt de taak teruggestuurd naar **IN PROGRESS** voor aanpassingen.

4. TESTFASE

- De taak wordt getest, en als de test succesvol is (**TESTING OK - RELEASE**), wordt de taak naar de releasefase gestuurd (**TO RELEASE**).
- Als de test niet slaagt (**Testing NOT OK - PICK UP AGAIN**), moet de taak opnieuw opgepakt en aangepast worden.

5. RELEASEFASE

- Als de release goedgekeurd is (**RELEASE OK**), kan de taak afgerond worden.
- Er is een optie om een nieuwsbrief te sturen (**SEND NEWSLETTER**) voordat de taak als **DONE** wordt gemarkeerd.

6. AFRONDING

- De taak eindigt in de **DONE**-status, wat betekent dat het proces volledig afgerond is.

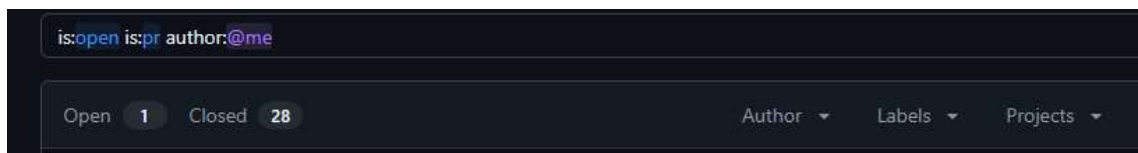
MOGELIJKE ROUTES:

- Taken kunnen meerdere keren herzien worden bij mislukte reviews of tests.
- Er is een optionele stap om een nieuwsbrief te sturen voordat een taak als afgerond wordt beschouwd.
- Ongeacht de status kan een taak in sommige gevallen direct als **DONE** worden gemarkeerd.

Ik start telkens met een aan mij toegewezen ticket van 'To DO' naar 'In Progress' te slepen. Elke feature request krijgt een OV-nummer. Dit vermeld ik altijd in de commit messages in Git, waardoor Jira automatisch de voortgang van het ticket bijhoudt. Na het programmeren maak ik een pull request en sleep ik het ticket naar "Tech Review" in Jira. De lead developer kijkt dit na en geeft feedback voor aanpassingen of verplaatst het ticket naar de testomgeving (merge met main). Vervolgens test QA de functionaliteit en geeft wederom feedback of verplaatst het naar 'To release' klaar om uitgebracht te worden naar de productie omgeving. Na dat de functionaliteiten Live worden gebracht, wordt het ticket afgesloten door het naar 'Done' te slepen.

3.3.2. Uitgelichte opgeloste tickets

Buiten de 2 grotere opdrachten droeg ik ook bij aan de dagelijkse ontwikkeling en kleinere aanpassingen. Ik loste in totaal 28 tickets op buiten de stage opdrachten.



Figuur 13: 28 gesloten tickets met aanpassingen.

OV-1488: MAAK TICKETS ONBEPERKT

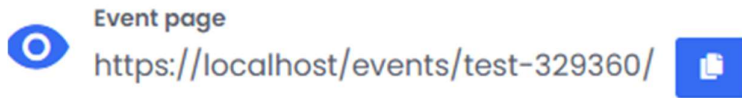
Bij het aanmaken van een tickettype was het verplicht om een aantal tickets op te geven. Dit veroorzaakte verwarring bij organisatoren die een onbeperkt aantal tickets wilden aanbieden. Ik paste het formulier aan zodat het veld optioneel werd, als het veld leeg blijft is het aantal beschikbare tickets onbeperkt. en voegde validatie toe om foutieve invoer te vermijden.

OV-1487: LIDMAATSCHAP VEREIST: DOORVERWIJZING NAAR LOGINPAGINA

Voor tickets die enkel beschikbaar zijn voor leden, werd aan niet-ingelogde gebruikers foutief de melding "No longer available" getoond. Dit gaf de indruk dat het ticket niet meer beschikbaar was, terwijl het in werkelijkheid enkel zichtbaar is na inloggen. Ik paste de frontend aan zodat in plaats van deze foutieve melding nu een duidelijke knop verschijnt met de tekst "Log in om dit ticket te bekijken". Deze knop verwijst rechtstreeks naar de loginpagina, wat de gebruikerservaring verbetert en verwarring voorkomt.

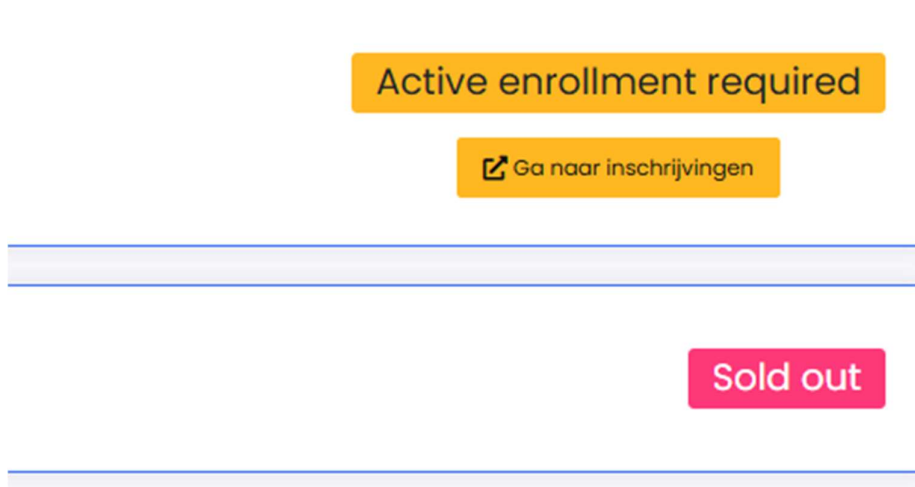
OV-1486: TOEVOEGEN VAN 'KOPIEER LINK'-KNOP OP EVENT DETAILPAGINA

Gebruikers wilden snel en eenvoudig de link naar een evenement kunnen delen. Daarom voegde ik een knop toe op de detailpagina van elk evenement waarmee de URL van het event automatisch naar het klembord wordt gekopieerd. Deze verbetering verhoogt de deelbaarheid van evenementen en draagt bij aan een vlottere gebruikerservaring.



OV-1472: LINK TOEVOEGEN NAAR INSCHRIJVINGSFORMULIER BIJ TICKETVEREISTE

Voor sommige tickets is een voorafgaande inschrijving vereist. Wanneer een gebruiker een dergelijk ticket probeert te selecteren zonder ingeschreven te zijn, werd er geen duidelijke actie voorgesteld. In dit ticket heb ik een knop toegevoegd die gebruikers rechtstreeks doorverwijst naar het inschrijvingsformulier. Hierdoor wordt de gebruikersflow intuïtiever en wordt vermeden dat gebruikers vastlopen zonder te weten wat de volgende stap is.



OV-1482: NIEUWE SUCCESPAGINA VOOR GRATIS TICKET

Wanneer een gebruiker een gratis ticket bestelde, werd hij na afronding van het proces doorgestuurd naar een standaard "betaling geslaagd"-pagina. Dit was verwarrend, aangezien er geen betaling had plaatsgevonden. In dit ticket implementeerde ik een aparte succespagina specifiek voor gratis tickets. Deze pagina bevestigt de inschrijving zonder te verwijzen naar een betaling, wat zorgt voor een duidelijkere en meer gepaste gebruikerservaring.

OV-1483: SUCCESMELDING BIJ GRATIS INSCHRIJVING

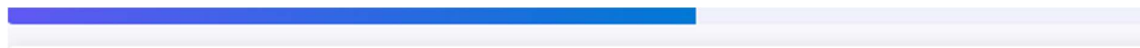
Wanneer een gebruiker zich inschreef voor een evenement waarvoor geen betaling vereist was, werd er geen duidelijke bevestiging getoond. Dit zorgde voor onzekerheid over de status van de inschrijving. In dit ticket implementeerde ik een specifieke succesmelding voor gratis inschrijvingen. Na succesvolle registratie wordt de gebruiker nu doorgestuurd naar een duidelijke bevestigingspagina, wat zorgt voor een betere gebruikerservaring en minder vragen bij de helpdesk.

OV-1495: VISUELE STATUSINDICATOR EN UITNODIGING BIJ LAGE LEDENAANTALLEN

In het onboardingproces voor clubs werd geen visuele feedback gegeven over het aantal leden. In dit ticket implementeerde ik een statusindicator die groen of rood kleurt afhankelijk van het aantal leden. Wanneer een club minder dan 10 leden heeft, verschijnt er automatisch een oproep onder de onboarding-links met een knop "Nodig leden uit". Dit helpt nieuwe clubs sneller op weg en verhoogt de betrokkenheid van beheerders bij het groeiproces van hun vereniging.



- **Publieke pagina** - Je publieke pagina is beschikbaar op <https://airsoft.ordolio.com/>
- **Registraties** - Nieuwe gebruikers kunnen zich registreren.
- **Inschrijvingen** - Je hebt geen actieve inschrijvingsperiodes. [Maak een inschrijvingsperiode aan](#)
- **Webshop** - Je webshop is nog niet actief. [Activeer je webshop](#)
- **Donaties** - Donaties zijn niet actief. [Waarom kan ik geen donaties ontvangen?](#)
- **Nodig je leden uit!** - Je hebt momenteel minder dan 10 leden. [Gebruikers uitnodigen](#)



- **Publieke pagina** - Je publieke pagina is beschikbaar op <https://airsoft.ordolio.com/>
- **Registraties** - Nieuwe gebruikers kunnen zich registreren.
- **Inschrijvingen** - Je hebt geen actieve inschrijvingsperiodes. [Maak een inschrijvingsperiode aan](#)
- **Webshop** - Je webshop is nog niet actief. [Activeer je webshop](#)
- **Donaties** - Donaties zijn niet actief. [Waarom kan ik geen donaties ontvangen?](#)
- **Nodig je leden uit!** - Je hebt momenteel meer dan 10 leden. [Gebruikers uitnodigen](#)



OV-1494: REDIRECT NAAR HOMEPAGE NA STRIPE-CONFIGURATIE

Na het succesvol instellen van de betalingsmodule via Stripe, werden gebruikers doorgestuurd naar een standaard webshopbericht dat niet relevant was voor hun context. Dit zorgde voor verwarring, vooral bij beheerders die geen webshop gebruikten. In dit ticket implementeerde ik een aangepaste redirect: na het voltooien van de Stripe-setup worden gebruikers nu automatisch teruggestuurd naar de homepage. Dit zorgt voor een logischere gebruikersflow en een meer consistente ervaring.

OV-1499: ACTIEVE STATUS VOOR INSCHRIJVINGSPERIODES

Om meer controle te bieden over wanneer gebruikers zich kunnen inschrijven, voegde ik een "actief"-vlag toe aan inschrijvingsperiodes. Standaard staat deze vlag op actief, wat betekent dat inschrijvingen mogelijk zijn, zelfs vóór de startdatum van de periode. Zodra de einddatum van de periode is bereikt, wordt inschrijving automatisch geblokkeerd. Indien de periode nog geldig is (bijvoorbeeld van januari tot december), maar de actief-vlag handmatig wordt uitgeschakeld, dan kunnen gebruikers zich ook niet meer inschrijven.

OV-1497: OVERZICHT ACTIEVE INSCHRIJVINGSPERIODES PER GROEP

Om beheerders een beter overzicht te geven van de inschrijvingsstatus per groep, voegde ik een nieuwe kolom toe aan het groepsbeheer. Deze kolom toont alle actieve inschrijvingsperiodes. Indien er geen actieve periodes zijn, wordt dit visueel aangeduid met een label in lichtrood ("No registration periods"). Daarnaast wordt er steeds een plus-icoon weergegeven. Wanneer dit icoon wordt aangeklikt, opent een modaal venster waarin de beheerder onmiddellijk een nieuwe inschrijvingsperiode kan aanmaken. Deze verbetering verhoogt de zichtbaarheid en het gebruiksgemak van het inschrijvingsbeheer.

OV-1257: STANDAARD E-MAILFOOTER INSTELBAAR PER ORGANISATIE

Om organisaties meer controle te geven over hun communicatie, implementeerde ik de mogelijkheid om een standaard e-mailfooter te definiëren. Deze footer wordt automatisch toegevoegd aan alle uitgaande e-mails die vanuit het platform worden verzonden. Organisaties kunnen de inhoud van deze footer zelf beheren via hun instellingen. Dit zorgt voor consistentere communicatie, verhoogt de herkenbaarheid van de afzender en maakt het mogelijk om bijvoorbeeld contactgegevens of disclaimers standaard mee te geven.

OV-1259: MOGELIJKHEID OM BIJLAGEN TOE TE VOEGEN AAN E-MAILS

Om communicatie via het platform te verrijken, implementeerde ik de mogelijkheid om bijlagen toe te voegen aan uitgaande e-mails. Beheerders kunnen nu eenvoudig bestanden uploaden die als echte e-mailbijlage worden meegestuurd, in plaats van enkel links te gebruiken. Dit is vooral nuttig voor het verzenden van documenten zoals inschrijvingsformulieren, facturen of informatiebrochures. De functionaliteit werd geïntegreerd in het bestaande e-mailsysteem en ondersteunt meerdere bestandstypes.

Compose Email

Here you can compose the email. Fill in the subject and body of the email and click Next.

Subject

Bijlage



Message

Here you can compose the content of the email. Use the editor to format the email.

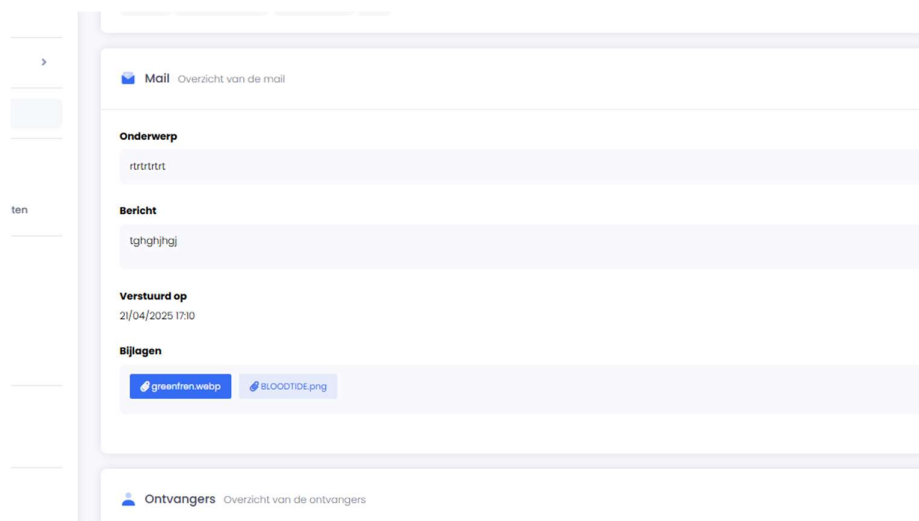
Please note that not every email client supports all formatting options. You can send a test email to yourself to make sure the formatting is displayed correctly.

[Send test email](#)[+ Add image](#)

File Edit View Insert Format Tools Table Help

OV-1646 – ONDERSTEUNING VOOR MEERDERE BIJLAGEN IN E-MAILS

Om de e-mailfunctionaliteit verder uit te breiden, implementeerde ik de mogelijkheid om meerdere bijlagen toe te voegen aan één e-mail. Voorheen kon slechts één bestand worden meegestuurd, wat beperkingen opleverde voor organisaties die bijvoorbeeld meerdere documenten tegelijk wilden verzenden. Dankzij deze uitbreiding kunnen gebruikers nu eenvoudig meerdere bestanden selecteren en toevoegen, die vervolgens als afzonderlijke bijlagen worden meegestuurd met de e-mail. Dit verhoogt de flexibiliteit en het professioneel gebruik van het platform.



4. Besluit

Dit realisatiedocument vormt de afsluiting van mijn stage bij Ordolio, waarin ik twee centrale opdrachten uitvoerde: het ontwikkelen van een wachtlijstfunctionaliteit en het uitvoeren van een penetratietest. Beide opdrachten boden mij de kans om mijn technische vaardigheden te verdiepen en praktijkervaring op te doen binnen een professionele ontwikkelomgeving.

De wachtlijstfunctionaliteit werd ontwikkeld op basis van een onderbouwd vergelijkend onderzoek. Door vier methodes te analyseren aan de hand van zes criteria binnen een Weighted Ranking Methode, kwam FIFO als meest geschikte oplossing naar voren. Deze methode werd succesvol geïmplementeerd als MVP binnen de bestaande infrastructuur van Ordolio.

De penetratietest werd opgestart naar aanleiding van een ontdekte XSS-kwetsbaarheid. De test werd grondig voorbereid en uitgevoerd volgens erkende methodologieën (OWASP Top 10, NIST SP 800-115) en resulteerde in concrete aanbevelingen voor het verbeteren van de beveiliging van de applicatie.

Daarnaast werkte ik mee aan de dagelijkse ontwikkeling van het platform en loste ik 26 tickets op, gaande van bugfixes tot UX-verbeteringen en nieuwe features.

Uit deze stage blijkt dat een gestructureerde aanpak, gecombineerd met technische diepgang en oog voor gebruiksvriendelijkheid, leidt tot duurzame oplossingen. Zowel de wachtlijstfunctionaliteit als de pentest droegen bij aan de kwaliteit en veiligheid van het platform.

De doelstellingen zoals geformuleerd in het plan van aanpak werden behaald:

- De wachtlijstfunctionaliteit werd niet alleen technisch gerealiseerd, maar ook onderbouwd met onderzoek.
- De penetratietest leverde waardevolle inzichten op en leidde tot concrete verbeteringen.
- Mijn bijdrage aan het team ging verder dan de hoofdopdrachten, wat mijn betrokkenheid en leercurve versterkte.

Op basis van deze ervaring formuleer ik enkele aanbevelingen:

- Voor de wachtlijstfunctionaliteit kan in de toekomst ondersteuning worden voorzien voor meerdere types (zoals prioriteitslijsten of loting).
- Voor de beveiliging raad ik aan om jaarlijks een pentest in te plannen en security-by-design structureel te integreren in het ontwikkelproces.
- Voor mezelf neem ik mee dat samenwerking, documentatie en testbaarheid minstens zo belangrijk zijn als de code zelf.

Deze stage was voor mij een waardevolle leerervaring die me niet alleen technisch, maar ook professioneel heeft doen groeien.

LITERATUURLIJST

Ordolio. (2025). *wij zijn Ordolio*. Opgehaald van Ordolio.com: <https://ordolio.com/wij-zijn-ordolio/>
Portswigger. (sd). *Cross-site Scripting*. Opgehaald van portswigger.net: <https://portswigger.net/web-security/cross-site-scripting>

BIJLAGEN

4.1. Bijlage A – Security Fixes

```
@@ -217,11 +219,13 @@ def format_periods_by_group_as_choices(periods_by_group: dict[Groepen, QuerySet[

    for group, periods in periods_by_group.items():
        for period in periods:

            if period.name:
                name = f'{group.naam} ({period.name})' if not html else f'<h4>
{group.naam}</h4> ({period.name})'
            else:
                name = f'{group.naam} (Periode: {period.get_period_display()})' if
not html else f'<h4>{group.naam}</h4> (Periode: {period.get_period_display()})'

            periods_by_group_choices.append((f'{period.id}_{group.id}', name))

219
220     for group, periods in periods_by_group.items():
221         for period in periods:
222 +         group_name = escape(strip_tags(group.naam))
223 +         period_name = escape(strip_tags(period.name)) if period.name else
period.get_period_display()
224
225         if period.name:
226 +         name = f'{group_name} ({period_name})' if not html else f'<h4>
{group_name}</h4> ({period_name})'
227         else:
228 +         name = f'{group_name} (Periode: {period_name})' if not html else
f'<h4>{group_name}</h4> (Periode: {period_name})'
229
230         periods_by_group_choices.append((f'{period.id}_{group.id}', name))
231
```

Figuur 14: Oplossing voor Cross site scripting

```
members/views.py
2 from dateutil.parser import parse
3 import json
4 from typing import Dict
5
6 from django.db.models.base import Model as Model
7 from django.shortcuts import render, redirect
8
9 @@ -2492,7 +2493,7 @@ def filter_queryset(self, qs: QuerySet[Inschrijvingen], filters: Dict[str, str])
10
11     def construct_period_html(self, registration: Inschrijvingen):
12         if self.request.current_organisation.use_multiple_groups:
13             - return f'{registration.groep.naam}<br/><small>{registration.period.get_name_or_period()}</small>'
14             + return f'{escape(registration.groep.naam)}<br/><small>{registration.period.get_name_or_period()}</small>'
15         else:
16             return f'{registration.period.get_name_or_period()}'
17
18 @@ -2512,7 +2513,7 @@ def construct_name_html(self, registration: Inschrijvingen):
19
20     <a href="{reverse('members:child_detail', args=[registration.kind.uuid])}">
21         <div class="d-flex align-items-center">
22             {registration_avatar}
23             - <span class="ml-3">{registration_name}</span>
24             + <span class="ml-3">{escape(registration_name)}</span>
25         </div>
26     </a>
27     ...
```

Figuur 15: Oplossing voor Cross site scripting