Plan van Aanpak Verbeteringen

Vibecode Quickscan

Blis Digital — Jorden Willemsen
1 april 2026

Datum: april 2026 **Auteur:** Blis Digital — Jorden Willemsen **Gebaseerd op:** 10-Pijler Vibecode Quickscan Assessment (maart 2026) *Dit is een geanonimiseerd voorbeeld van een werkelijk opgeleverd plan van aanpak, volgend op een Vibecode Quickscan. Klantnaam, domeinen en klantspecifieke koppelingen zijn verwijderd of vervangen door generieke aanduidingen.*

---

1. Aanleiding

Uit de 10-Pijler Review (maart 2026) blijkt dat de applicatie een sterke verbetering heeft doorgemaakt: van **1.75/5** (eerdere externe review, februari 2026) naar **3.3/5**. De kern van de applicatie — beveiliging, authenticatie, privacy en code-kwaliteit — scoort goed (4/5). Echter, twee pijlers scoren onvoldoende en vormen de grootste blokkeerders voor duurzaam, schaalbaar beheer:

Pijler	Score	Kernprobleem
Testen	2/5	Geen frontend-tests, geen integratietests, geen automatische testrun
Ontwikkelproces	2/5	Geen CI/CD-pipeline, geen staging-omgeving, geen PR-workflow

Zonder verbetering op deze twee pijlers is doorontwikkelen risicovol: regressies worden niet automatisch gedetecteerd, deploys zijn niet reproduceerbaar en er is geen vangnet bij fouten.

---

2. Doelstelling

Streefniveau: alle 10 pijlers op minimaal 3/5, met Testen en Ontwikkelproces op 4/5.
Verwacht resultaat: een totaalscore van minimaal **4.0/5**, waarmee de applicatie niet alleen productiewaardig is maar ook duurzaam onderhoudbaar en veilig doorontwikkelbaar.

---

3. Scope en afbakening

Dit plan richt zich op de verbeterpunten uit de 10-Pijler Review. Het is opgedeeld in drie fasen:

- **Fase 1 — Direct oppakken** (week 1-2): hoge-prioriteit blokkeerders
- **Fase 2 — Plannen en inrichten** (week 3-6): middel-prioriteit verbeteringen
- **Fase 3 — Doorontwikkelen** (week 7-12): langere-termijn kwaliteitsverbetering

Buiten scope: functionele doorontwikkeling, UI-redesign, migratie naar andere platformen.

---

4. Fase 1 — Direct oppakken (week 1-2)

4.1 CI/CD-pipeline instellen (HOOG)

Probleem: Er is geen geautomatiseerde pipeline. Deploys zijn handmatig en niet reproduceerbaar. Tests draaien niet automatisch. **Aanpak:**

- GitHub Actions workflow instellen met de volgende stappen:
  - `npm ci` — dependencies installeren
  - `npm run lint` — ESLint check
  - `npm run build` — build-validatie
  - `npm run test:all` — alle bestaande tests (Cloud Functions + Firestore rules)
- Trigger: bij elke push naar `main` en bij pull requests
- Deploy naar Firebase Hosting na succesvolle build op `main`
- Firestore rules en Cloud Functions deployen als onderdeel van dezelfde pipeline

Resultaat: elke codewijziging wordt automatisch gevalideerd. Broken builds komen niet meer in productie. **Impact op score:** Ontwikkelproces van 2/5 naar 3/5

4.2 npm audit + react-router CVE-verificatie (HOOG)

Probleem: De eerdere externe review meldde CVEs (XSS, CSRF) in react-router. Het is niet geverifieerd of v7.9.4 deze verhelpt. **Aanpak:**

- `npm audit` draaien op zowel de frontend als `functions/` directory
- Specifiek verifiëren dat react-router-dom v7.9.4 de eerder gemelde CVEs heeft gepatcht
- Eventuele overige kwetsbaarheden documenteren en prioriteren

- `npm audit` opnemen als stap in de CI/CD-pipeline (zie 4.1)

Resultaat: bevestiging dat bekende kwetsbaarheden zijn verholpen, en continue bewaking via CI. **Impact op score:** Beveiliging blijft 4/5, Licenties & Dependencies van 3/5 naar 4/5

---

5. Fase 2 — Plannen en inrichten (week 3-6)

5.1 Firebase API-sleutel restrictie (MIDDEL)

Probleem: De publieke Firebase API-sleutel in `firebase.js` is onbeperkt. Misbruik is mogelijk als de sleutel buiten de beoogde context wordt gebruikt. **Aanpak:**

- In Google Cloud Console de API-sleutel beperken tot het productiedomein
- HTTP Referrer-restrictie instellen
- Documenteren in de projectdocumentatie welke restricties zijn toegepast

Resultaat: verkleind aanvalsoppervlak voor misbruik van de Firebase API-sleutel.

5.2 Dode dependency nodemailer verwijderen (MIDDEL)

Probleem: `nodemailer` staat in `functions/package.json` maar wordt nergens gebruikt. E-mail gaat via de API van de e-mailprovider (Axios). De dependency vergroot het aanvalsoppervlak onnodig. **Aanpak:**

- `nodemailer` verwijderen uit `functions/package.json`
- `npm install` draaien om de lockfile bij te werken
- Verificatie dat alle Cloud Functions nog correct werken
- Bevestigen via CI-pipeline

Resultaat: verkleind aanvalsoppervlak, minder onderhoudslast.

5.3 Auth + Storage backups uitbreiden (MIDDEL)

Probleem: De dagelijkse Firestore-backup dekt alleen Firestore-collecties. Firebase Auth-gebruikers en Storage-bestanden (beeldbank) worden niet geback-upt. Bij verlies van het Firebase-project zijn deze gegevens weg. **Aanpak:**

- Periodieke export van Firebase Auth-gebruikers via `firebase auth:export` of Admin SDK
- Firebase Storage-bestanden synchroniseren naar een apart Cloud Storage-bucket

- Beide opnemen in een scheduled Cloud Function (wekelijks)
- Disaster recovery-procedure documenteren en eenmalig testen

Resultaat: volledige backup-dekking en een getest herstelproces. **Impact op score:** Back-ups van 3/5 naar 4/5

---

6. Fase 3 — Doorontwikkelen (week 7-12)

6.1 Frontend-tests toevoegen

Probleem: Er zijn geen frontend-tests. Het test-framework ontbreekt in de frontend `package.json`. **Aanpak:**

- Vitest installeren als test-framework (sluit aan bij Vite-configuratie)
- React Testing Library toevoegen voor component-tests
- Tests schrijven voor kerncomponenten: `AdminRoute`, `CookieBanner`, registratieformulier, login-flow
- Minimale coverage-drempel instellen (bijv. 60%) en opnemen in CI

Resultaat: regressiebeveiliging op frontend-niveau. **Impact op score:** Testen van 2/5 naar 3/5

6.2 Staging-omgeving opzetten

Probleem: Wijzigingen gaan direct naar productie. Er is geen mogelijkheid om te testen in een productie-achtige omgeving zonder eindgebruikers te raken. **Aanpak:**

- Tweede Firebase-project aanmaken als staging-omgeving
- CI/CD-pipeline uitbreiden: feature-branches deployen naar staging, `main` naar productie
- Firestore rules, Cloud Functions en hosting spiegelen naar staging
- Testdata aanmaken voor de staging-omgeving

Resultaat: veilige testomgeving voor nieuwe features en hotfixes. **Impact op score:** Ontwikkelproces van 3/5 naar 4/5, Hosting van 4/5 naar 5/5

6.3 Proactieve alerting en monitoring

Probleem: Sentry is geïntegreerd maar er is geen proactieve alerting. Bij crashes of kritieke fouten wordt niemand automatisch gewaarschuwd. **Aanpak:**

- Sentry-alerts configureren voor nieuwe errors en error-spikes
- Uptime-monitoring instellen (bijv. Firebase Hosting health check of externe service)
- Sentry DSN verplaatsen van hardcoded waarde in `main.jsx` naar environment variable
- Budget-alerts instellen in Google Cloud Console voor onverwachte kostenpieken

Resultaat: proactieve waarschuwingen bij problemen, vóórdat gebruikers het melden. **Impact op score:** Monitoring van 3/5 naar 4/5

6.4 PR-workflow en code-review proces

Probleem: Er is geen formeel code-review-proces of PR-workflow. **Aanpak:**

- Branch protection rules instellen op `main`: minimaal 1 review vereist
- PR-template toevoegen met checklist (tests, lint, security check)
- CI verplicht laten slagen vóór merge
- Deploy-procedure documenteren

Resultaat: gestructureerd ontwikkelproces met kwaliteitscontrole. **Impact op score:** Ontwikkelproces naar 4/5

6.5 Bewaarbeleid en opruiming login-logs

Probleem: Login-logs (`logs_login`) bevatten `userAgent` (PII) en er is geen bewaarbeleid. **Aanpak:**

- Maximale bewaartermijn vaststellen (bijv. 90 dagen) in overleg met de klant
- Cloud Function schedulen die logs ouder dan de bewaartermijn automatisch verwijdert
- Privacybeleid bijwerken met documentatie over login-logging

Resultaat: AVG-conforme logverwerking.

---

7. Verwachte scoreverbeteringen

Pijler	Huidig	Na fase 1	Na fase 2	Na fase 3	Streef
1. Authenticatie & Autorisatie	4/5	4/5	4/5	4/5	4/5
2. Beveiliging	4/5	4/5	4/5	4/5	4/5
3. Privacy & AVG	4/5	4/5	4/5	4/5	4/5
4. Code Kwaliteit	4/5	4/5	4/5	4/5	4/5
5. Testen	2/5	2/5	2/5	4/5	4/5
6. Hosting	4/5	4/5	4/5	5/5	5/5
7. Back-ups	3/5	3/5	4/5	4/5	4/5
8. Licenties & Dependencies	3/5	4/5	4/5	4/5	4/5
9. Monitoring	3/5	3/5	3/5	4/5	4/5
10. Ontwikkelproces	2/5	3/5	3/5	4/5	4/5
Totaal	**33/50**	**35/50**	**36/50**	**41/50**	**41/50**
Gemiddeld	**3.3/5**	**3.5/5**	**3.6/5**	**4.1/5**	**4.1/5**

---

8. Randvoorwaarden en risico's

Randvoorwaarde	Toelichting
Toegang tot GitHub repository	CI/CD-pipeline vereist een repository met GitHub Actions
Toegang tot Google Cloud Console	Nodig voor API-sleutel restricties, budget-alerts en backup-configuratie
Toegang tot Firebase Console	Voor staging-project, Auth export, en monitoring-configuratie
Medewerking klantteam	Afstemming over bewaarbeleid, testdata en acceptatie-criteria

Risico	Mitigatie
CI/CD-pipeline vertraagt deploy-snelheid	Pipeline lean houden; alleen essentiële checks
Staging-omgeving brengt extra kosten	Firebase Spark (gratis) plan gebruiken voor staging
Frontend-tests vergen initieel veel tijd	Focussen op kernflows, niet op 100% coverage

---

9. Samenvatting

De applicatie is productiewaardig (3.3/5) maar nog niet duurzaam onderhoudbaar. Dit plan van aanpak richt zich op de twee zwakste pijlers — **Testen** en **Ontwikkelproces** — en brengt de totaalscore naar **4.1/5** in drie fasen over 12 weken.

De eerste twee weken leveren het grootste rendement: een werkende CI/CD-pipeline en geverifieerde dependency-security. De vervolgfasen bouwen daarop voort met frontend-tests, een staging-omgeving, proactieve monitoring en een formeel review-proces.

Met dit plan groeit de applicatie van een productieklaar prototype naar een robuust, schaalbaar en professioneel onderhoudbaar platform.



Unleash the power of technology

www.blisdigital.com