Techspike

Audit de sécurité et de code

Application **Portail client** · Node.js, Express, TypeScript, SQLite

ÉTABLI POUR

Muster AG (fictive)

Max Beispiel, Responsable informatique

VOTRE INTERLOCUTEUR

Benjamin Barbé

Date	Période d'audit	Version	Classification
1er novembre 2026	1er novembre 2026	1.0	Exemple, données fictives

Techspike SARL
39 rue du Général de Gaulle
33470 Margaux-Cantenac
France

Sommaire

1 Synthèse 2

1.1 Mission et périmètre 2

1.2 Évaluation globale 3

1.3 Évaluation par domaine 4

1.4 Actions prioritaires 5

2 Plan d'action 6

3 Constats de sécurité 8

4 Qualité, exploitation et dépendances 25

4.1 Qualité du code et maintenabilité 25

4.2 Complexité et volume 28

4.3 Performance et cache 31

4.4 Tests 34

4.5 Build, CI/CD et exploitation 35

4.6 Dépendances 37

5 Analyse automatisée 37

6 Périmètre, méthodologie et limites 39

A Annexe 41

CHAPITRE 1

Synthèse

1.1 Mission et périmètre

Ce rapport est un exemple. La Muster AG et son portail client sont fictifs ; l'application a été écrite par Techspike spécialement comme objet d'audit et contient délibérément des vulnérabilités typiques. Déroulement, méthodologie et mise en page correspondent à un véritable audit de code.

L'objet de l'audit est le portail client, par lequel les clients professionnels consultent commandes et factures, déposent des documents et configurent des webhooks pour leur ERP. L'audit a porté sur l'intégralité du code source, historique Git compris, ainsi que sur la configuration de build et de déploiement, dans l'état de la branche principale.

L'audit était statique et en lecture seule : analyse automatisée des secrets, des vulnérabilités connues, des motifs dangereux et des erreurs de configuration, puis revue assistée par IA pour chaque domaine. Chaque constat de gravité élevée ou plus a été contre-vérifié par un auditeur indépendant avant d'être intégré à ce rapport.

Client	Muster AG (fictive) · Max Beispiel, Responsable informatique
Prestataire	Techspike SARL · Benjamin Barbé
Référence	Rapport d'exemple – application fictive
Projet	Portail client · Node.js, Express, TypeScript, SQLite · accessible publiquement
Dépôt Git	`muster-kundenportal (bundle Git)` · Branche main · Commit `e0e3cde972c5` · 20/10/2026
Volume	22 fichiers · 309 lignes de code de production · 30 lignes de code de test
Période	1er novembre 2026 · À distance, poste d'audit de Techspike SARL
Type d'audit	Statique, en lecture seule · profondeur « standard » · pas de test d'intrusion
Référentiels	Outils (secrets, CVE, SAST, configuration) et revue assistée par IA selon l'OWASP Top 10:2025 ; cotation CVSS 3.1, correspondance CWE
Modèle d'IA	Claude Opus 5.5 · `claude-opus-5-5` 8 agents spécialisés d'audit et de revue ; chaque constat pertinent contre-vérifié et validé par un agent indépendant

1.2 Évaluation globale

APPRÉCIATION GLOBALE

Insuffisant

Niveau 1 sur 4 · 1,50 points

En l'état, le portail n'est pas apte à une exploitation sur Internet : n'importe qui peut s'octroyer des droits d'administrateur sans identifiants et accéder ainsi aux données de tous les clients.

Échelle : 4 très bon · 3 bon · 2 actions requises · 1 insuffisant. La sécurité compte pour 40 % ; si elle est insuffisante, l'appréciation globale l'est aussi. Échelle et pondération au chapitre Méthodologie

Le portail client est sobre et lisible. Sa sécurité repose toutefois sur des contrôles écrits à la main dans chaque route – et absents à plusieurs endroits. Le point le plus grave est l'inscription : rôle et numéro client proviennent de la requête sans contrôle. Qui le sait devient administrateur en quelques minutes.

S'y ajoutent des failles qui permettent à des clients connectés de lire factures, fichiers et contenus de base de données d'autres clients, une réinitialisation de mot de passe sans aucune preuve et une clé de signature inscrite dans le code. La plupart des mesures immédiates sont des modifications locales. Les failles critique et élevées se corrigent en trois jours de développement environ, l'ensemble des mesures immédiates en sept à dix.

Sur le plan structurel manquent un contrôle avant déploiement, des tests du cloisonnement des clients et une routine de mise à jour des dépendances. Sans ces trois éléments, les mêmes erreurs réapparaîtront à la prochaine évolution.

1.3 Évaluation par domaine

DOMAINE	ÉVALUATION	EN BREF	PAGE
Sécurité Constats : 1 C · 5 É · 17 M · 7 F	Insuffisant	Une faille critique à l'inscription et plusieurs accès aux données d'autres clients rendent le cloisonnement des clients inopérant.	8 →
Qualité du code et maintenabilité Constats : 4 M · 1 F	Actions requises	Le code est court et lisible, mais l'accès aux données, la gestion des erreurs et le typage ne sont pas traités de façon homogène.	25 →
Complexité et volume Constats : 1 M · 2 F	Bon	Les fonctions sont simples et sans duplication ; la sur-ingénierie se limite à un module central bien délimité.	28 →
Performance et cache Constats : 2 M · 2 F	Actions requises	Le seul cache n'apporte aucun gain de vitesse, mais sert des factures périmées et appartenant à d'autres clients.	31 →
Tests Constats : 1 É	Insuffisant	Les tests sont ponctuels ; la logique de sécurité et de cloisonnement n'est pas testée.	34 →
Build, CI/CD et exploitation Constats : 1 É · 3 M · 3 F	Actions requises	Le déploiement est automatisé, mais part directement en production sans contrôles et sur un environnement d'exécution qui n'est plus maintenu.	35 →
Dépendances Constats : 1 É · 2 M	Insuffisant	Six des neuf bibliothèques d'exécution ont une ou plusieurs versions majeures de retard et présentent des vulnérabilités connues ; aucune routine de mise à jour n'existe.	37 →

1.4 Actions prioritaires

SEC-01	Ignorer `role` à l'inscription, rattachement client uniquement sur invitation	CRITIQUE	8
SEC-02	Réinitialisation uniquement par lien e-mail unique et limité dans le temps	ÉLEVÉ	9
SEC-03	Intercepter les erreurs du handler de réinitialisation, valider `email`	ÉLEVÉ	9
SEC-04	Mettre à jour Express (5.x) et Multer (≥ 2.3.0)	ÉLEVÉ	10
SEC-05	Supprimer le secret de repli, régénérer la clé JWT	ÉLEVÉ	10

Les 34 actions immédiates et les améliorations structurelles : Plan d'action, page 6

POINTS FORTS

+ Les fonctions sont petites et claires, sans duplication de code ni imbrication complexe.
+ Toutes les zones non publiques exigent une authentification au niveau du routeur, l'administration en plus un contrôle de rôle ; hormis la recherche de commandes, toutes les requêtes utilisent des paramètres liés.
+ Les dépendances sont versionnées exactement et figées par un lockfile, `.env` est exclu du versionnement – une bonne base pour des builds reproductibles ; le pipeline n'utilise toutefois pas encore le lockfile.

CHAPITRE 2

Plan d'action

Les actions immédiates corrigent des failles précises et doivent être déployées avant la prochaine mise en production. Les améliorations structurelles réduisent durablement le risque et relèvent de la planification.

ACTIONS IMMÉDIATES

ID	ACTION	GRAVITÉ	PAGE
SEC-01	Ignorer `role` à l'inscription, rattachement client uniquement sur invitation	CRITIQUE	8
SEC-02	Réinitialisation uniquement par lien e-mail unique et limité dans le temps	ÉLEVÉ	9
SEC-03	Intercepter les erreurs du handler de réinitialisation, valider `email`	ÉLEVÉ	9
SEC-04	Mettre à jour Express (5.x) et Multer (≥ 2.3.0)	ÉLEVÉ	10
SEC-05	Supprimer le secret de repli, régénérer la clé JWT	ÉLEVÉ	10
SEC-06	Limitation de débit et frein aux échecs à la connexion, règle de mot de passe renforcée	ÉLEVÉ	11
QA-01	Placer un job de contrôle (typage, lint, tests) avant le déploiement	ÉLEVÉ	35
SEC-07	Livrer le lockfile et utiliser `npm ci` au lieu de `npm install` sur le serveur	MOYEN	11
SEC-08	Téléchargement uniquement par ID de document avec filtre client, chemin limité au répertoire de téléversement	MOYEN	12
SEC-09	Requête paramétrée dans `GET /api/orders/search`	MOYEN	13
SEC-10	Filtre client dans `GET /api/invoices/:id`, supprimer le cache des factures	MOYEN	13
SEC-11	Limitation de débit et confirmation d'adresse pour l'inscription et la réinitialisation	MOYEN	14
SEC-12	Filtrer les URL de webhook vers des cibles internes, ne pas renvoyer le contenu de la réponse	MOYEN	15
SEC-13	Activer la vérification des certificats et `requireTLS` dans le transport SMTP	MOYEN	15
SEC-14	Limite du corps à 100 ko, limites de téléversement dans `multer`	MOYEN	16
SEC-15	Stocker les fichiers sous un nom aléatoire par client, limiter taille et type	MOYEN	16
SEC-16	Passer `errorHandler` à un message générique avec ID d'erreur	MOYEN	17
SEC-18	Réponses uniformes à la connexion, à la réinitialisation et à l'inscription	MOYEN	18

ID	ACTION	GRAVITÉ	PAGE
SEC-19	Mettre à jour nodemailer, axios et lodash, activer Renovate	MOYEN	19
SEC-20	Supprimer la journalisation du corps de requête à la connexion, effacer les anciens journaux	MOYEN	19
SEC-21	Révoquer et renouveler les secrets de l'historique, gitleaks dans la CI	MOYEN	20
SEC-23	jsonwebtoken 9.x, algorithme fixé dans `verifyToken`	MOYEN	21
QA-04	Charger les lignes en lot, index sur `order_items.order_id`	MOYEN	31
QA-08	Journaliser les erreurs ignorées, découpler l'envoi de courriels, sécuriser les handlers async	MOYEN	26
QA-10	Supprimer `invoiceCache`, lire la facture directement avec filtre client	MOYEN	32
SEC-24	Dockerfile : `USER node`, `.dockerignore`, multi-étapes avec `npm ci`	FAIBLE	21
SEC-25	Déclarer la clé d'hôte dans `known_hosts`, supprimer `StrictHostKeyChecking=no`	FAIBLE	22
SEC-26	Épingler les actions par SHA, `permissions: contents: read`, environnement de déploiement	FAIBLE	22
SEC-27	Lire le jeton uniquement dans l'en-tête Authorization	FAIBLE	23
SEC-28	Remplacer le repli CORS `true` par une liste fixe d'origines autorisées	FAIBLE	23
SEC-29	Intégrer `helmet()`, retirer la version de `/health`	FAIBLE	24
SEC-30	Retirer l'empreinte de mot de passe de la liste des utilisateurs de l'administration	FAIBLE	24
QA-14	Supprimer l'endpoint de réglages sans effet et Lodash, ou réaliser un vrai enregistrement	FAIBLE	31
QA-15	Index sur `customer_id` (pour les commandes, avec `created_at`)	FAIBLE	33

AMÉLIORATIONS STRUCTURELLES

ID	ACTION	GRAVITÉ	PAGE
QA-02	Bâtir une base de tests avec clients de test, tester cloisonnement et authentification par route	ÉLEVÉ	34
SEC-17	Passer le hachage des mots de passe à Argon2id, migrer l'existant	MOYEN	18
SEC-22	Expiration fixe des jetons, révocation par version de jeton par utilisateur	MOYEN	20
QA-03	Mettre en place Dependabot/Renovate, rattraper progressivement les versions majeures	MOYEN	35
QA-05	Node LTS. `npm ci` avec lockfile. un seul circuit d'artefact de build	MOYEN	36

ID	ACTION	GRAVITÉ	PAGE
QA-06	Regrouper l'accès aux données par domaine, client en paramètre obligatoire, migrations	MOYEN	25
QA-07	Validation par schéma sur toutes les routes, abaisser la limite JSON	MOYEN	26
QA-09	Remplacer classe de base et fabrique du repository par deux fonctions de données	MOYEN	29
QA-11	Journal d'audit pour connexion, réinitialisation et actions d'administration	MOYEN	27
QA-12	Types de ligne par table, entrées validées par schéma, `any` interdit par lint	FAIBLE	28
QA-13	Remplacer le bus d'événements par un appel direct avec journalisation des erreurs	FAIBLE	30
QA-16	Pagination des listes, export en flux	FAIBLE	33

CHAPITRE 3

Constats de sécurité

Chaque constat a été vérifié dans le code et coté selon CVSS 3.1. Classement par gravité.

SEC-01

N'importe qui peut s'inscrire comme administrateur ou dans le compte d'un autre client

CRITIQUE

CVSS 9,1 `AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N` · OWASP A01:2025 Broken Access Control · CWE-269 · Authentification · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

```
src/routes/auth.ts:11
src/routes/auth.ts:20
src/middleware/auth.ts:28
```

CONSTAT

`POST /api/auth/register` est accessible sans authentification et reprend `role` et `customerId` tels quels depuis la requête (`auth.ts:11`, `:20`). Il n'y a ni invitation, ni contrôle du numéro client, ni confirmation de l'adresse e-mail.

RISQUE

Un attaquant anonyme s'inscrit avec `role: "admin"` et obtient via `/api/admin/*` tous les utilisateurs avec leurs empreintes de mots de passe, tous les clients et l'export. Il peut aussi se rattacher à n'importe quel numéro client et lire ses commandes, factures et documents.

CORRECTIF

Fixer le rôle côté serveur à `user` et ignorer `role` dans la requête. N'autoriser le rattachement à un client que par une invitation unique et limitée dans le temps, émise par un administrateur (rôle `admin`), et n'activer le compte qu'après confirmation de l'adresse e-mail.

SEC-01 · suite · N'importe qui peut s'inscrire comme administrateur ou dans le compte d'un autre client

EXTRAIT

```
const { email, password, customerId, role } = req.body;
...
  .run(customerId, email, hashPassword(password), role || 'user');
```

SEC-02

Le mot de passe de tout compte peut être réinitialisé sans preuve, puis deviné

ÉLEVÉ

CVSS 8,1 `AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H` · OWASP A07:2025 Authentication Failures · CWE-640 · Authentification · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/auth.ts:47`
`src/routes/auth.ts:53`
`src/routes/auth.ts:54`
`src/routes/auth.ts:55`

CONSTAT

`POST /api/auth/password-reset` remplace immédiatement le mot de passe de n'importe quel compte par un code à six chiffres, sans preuve que la demande émane du titulaire (`auth.ts:54`). Le code est issu de `Math.random` (`:53`) et, contrairement à ce qu'indique la réponse (`:55`), n'est envoyé nulle part.

RISQUE

Un attaquant anonyme bloque l'accès de n'importe quel utilisateur, administrateurs compris. La connexion ne limitant pas les tentatives, il peut ensuite essayer le million de codes possibles et prendre le contrôle du compte ; le dépôt ne montre pas si nginx freine ces essais.

CORRECTIF

Ne changer le mot de passe qu'après confirmation par l'utilisateur d'un lien envoyé par e-mail. Générer pour cela un jeton avec `crypto.randomBytes(32)` , ne le stocker que haché, le limiter à environ 30 minutes et à un seul usage, et limiter les demandes par adresse et par IP.

EXTRAIT

```
const code = String(Math.floor(Math.random() * 1000000)).padStart(6, '0');
db.prepare('UPDATE users SET password_hash = ? WHERE id = ?').run(hashPassword(code), user.id);
```

SEC-03

Plantage du portail par un seul appel anonyme à la réinitialisation du mot de passe

ÉLEVÉ

CVSS 7,5 `AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H` · OWASP A10:2025 Mishandling of Exceptional Conditions · CWE-248 · Gestion des erreurs · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/auth.ts:47`
`src/routes/auth.ts:49`

CONSTAT

`POST /api/auth/password-reset` est un handler `async` sans `try/catch` . Un corps comme `{"email": true}` fait lever une exception à better-sqlite3 lors de la liaison du paramètre ; sous Express 4, elle devient un rejet de promesse non géré, qui met fin au processus sous Node.js 16.

SEC-03 · suite · Plantage du portail par un seul appel anonyme à la réinitialisation du mot de passe

RISQUE Un attaquant anonyme paralyse le portail avec une seule requête et peut recommencer après chaque redémarrage.

CORRECTIF Entourer le handler d'un `try/catch` et transmettre les erreurs à `next` (ou utiliser Express 5, qui gère lui-même les promesses rejetées) ; valider au préalable `email` comme chaîne de caractères.

EXTRAIT

```
authRouter.post('/password-reset', async (req, res) ⇒ {
  const { email } = req.body;
  const user = db.prepare('SELECT id FROM users WHERE email = ?').get(email) as any;
```

SEC-04

Portail paralysable sans authentification par une requête forgée (Express obsolète)

ÉLEVÉ

CVSS 7,5 `AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H` · OWASP A03:2025 Software Supply Chain Failures · CWE-1395 · Dépendances · Source : osv-scanner · ↑ Évaluation par domaine

EMPLACEMENT
`package.json:17`
`src/server.ts:15`
`package-lock.json (qs 6.7.0)`

CONSTAT Express 4.17.1 embarque body-parser 1.19.0 et qs 6.7.0. Deux voies sont accessibles sans authentification : une seule requête JSON au corps gzip suivi d'octets supplémentaires fait planter body-parser (CVE-2026-111181, `express.json()` s'exécute pour chaque requête), et des chaînes de requête forgées bloquent le processus via qs (CVE-2022-24999).

RISQUE Un attaquant anonyme fait planter le processus Node avec une seule requête forgée ou le bloque par une chaîne de requête ; le portail devient alors inaccessible à tous les clients.

CORRECTIF Passer Express à la version 5.x actuelle (au minimum une 4.x qui résout qs ≥ 6.16.1, body-parser ≥ 1.20.9 et path-to-regexp ≥ 0.1.13) et Multer à ≥ 2.3.0 ; intégrer ensuite au pipeline un contrôle `npm audit` sans résultat sur les paquets d'exécution.

SEC-05

Jetons d'authentification falsifiables si la clé inscrite dans le code est utilisée

ÉLEVÉ

CVSS 7,4 `AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N` · OWASP A04:2025 Cryptographic Failures · CWE-321 · Cryptographie · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT
`src/config.ts:7`
`src/auth/jwt.ts:16`
`src/auth/jwt.ts:20`

SEC-05 · suite · Jetons d'authentification falsifiables si la clé inscrite dans le code est utilisée

CONSTAT Si `JWT_SECRET` n'est pas défini, l'application signe et vérifie les jetons avec une clé écrite en dur dans le code (`config.ts:7`) ; rien n'empêche un démarrage sans secret. La même valeur figurait aussi dans le `.env` du premier commit ; elle est donc connue de quiconque a accès au code ou à l'historique.

RISQUE Si la production tourne avec cette valeur – ce qui sort du périmètre du dépôt –, un attaquant anonyme forge un jeton avec `role: "admin"` et un `customerId` quelconque et obtient un accès complet en lecture et en écriture à tous les clients.

CORRECTIF Supprimer la valeur de repli et interrompre le démarrage si `JWT_SECRET` manque ou fait moins de 32 octets. En production, définir une nouvelle clé aléatoire (par ex. `openssl rand -base64 48`) ; tous les jetons existants deviennent alors invalides.

EXTRAIT

```
jwtSecret: process.env.JWT_SECRET || '<im Code hinterlegter Wert>',
```

SEC-06

Mots de passe testables sans limite à la connexion

ÉLEVÉ

CVSS 7,4 `AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N` · OWASP A07:2025 Authentication Failures · CWE-307 · Authentification · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT
`src/routes/auth.ts:28`
`src/auth/password.ts:13`
`src/server.ts:19`

CONSTAT `POST /api/auth/login` n'applique ni limitation de débit, ni blocage, ni délai après des échecs (`auth.ts:28`) ; aucune limitation n'est intégrée dans l'application (`server.ts:19`). La seule règle de mot de passe est une longueur minimale de six caractères (`password.ts:13`).

RISQUE Un attaquant anonyme teste des mots de passe courants ou issus d'autres fuites sur les adresses des clients et prend le contrôle des comptes à mot de passe faible. Une limitation par nginx sort du périmètre du dépôt ; d'où le classement en exploitation plus difficile.

CORRECTIF Limiter les échecs par compte et par IP, par ex. avec `express-rate-limit` et un délai progressif. Porter la longueur minimale à 12 caractères, comparer les mots de passe à une liste de mots de passe divulgués et prévoir un second facteur pour les comptes administrateurs.

SEC-07

Le serveur de production installe les dépendances sans lockfile à chaque déploiement

MOYEN

CVSS 6,8 `AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:N` · OWASP A03:2025 Software Supply Chain Failures · CWE-829 · CI/CD et chaîne d'approvisionnement · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT
`.github/workflows/deploy.yml:21`
`.github/workflows/deploy.yml:22`

SEC-07 · suite · Le serveur de production installe les dépendances sans lockfile à chaque déploiement

CONSTAT

Le pipeline ne transfère sur le serveur que `dist` et `package.json`, sans `package-lock.json`, puis y exécute `npm install --production`. Les paquets directs sont certes épinglés, mais toutes les dépendances transitives sont résolues à nouveau en production à chaque déploiement et leurs scripts d'installation exécutés – avec des versions ni construites ni testées.

RISQUE

Si un paquet npm transitif est compromis (comme lors de plusieurs incidents en 2025), son code malveillant s'exécute au push suivant sur `main` directement sur le serveur de production, avec les droits de l'utilisateur de déploiement, et atteint base, fichiers et configuration. Cela suppose un paquet compromis au moment du déploiement, d'où une complexité d'attaque élevée.

CORRECTIF

Ne plus installer sur le serveur : construire dans le job CI un artefact complet (`node_modules` compris) avec `npm ci --omit=dev` et le transférer, ou au minimum copier `package-lock.json` et utiliser `npm ci --omit=dev --ignore-scripts` sur le serveur (en compilant alors les modules natifs comme better-sqlite3). Utiliser aussi `npm ci` au lieu de `npm install` en CI.

EXTRAIT

```
rsync -e "ssh -i key ..." -r dist package.json deploy@portal.muster-ag.example:/srv/portal
ssh -i key ... "cd /srv/portal && npm install --production && pm2 restart portal"
```

SEC-08

Fichiers quelconques du serveur et documents d'autres clients téléchargeables

MOYEN

CVSS 6,5 `AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N` · OWASP A01:2025 Broken Access Control · CWE-22 · Entrées et fichiers · Source : semgrep · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/documents.ts:36`
`src/routes/documents.ts:37`
`src/routes/documents.ts:41`

CONSTAT

`GET /api/documents/download` accole le paramètre `file` au répertoire de téléversement via `path.join` et sert le fichier. Rien ne vérifie que le chemin reste dans ce répertoire (`../` est résolu), ni que le fichier appartient au client de l'utilisateur ; la table `documents` n'est même pas consultée.

RISQUE

Un utilisateur connecté télécharge avec `?file=../data/portal.db` la base de données de tous les clients (empreintes de mots de passe comprises), avec `?file=../.env` les secrets du serveur, ou tout fichier lisible par le processus. Même sans traversée, un nom de fichier connu ou deviné suffit pour obtenir les documents d'autres clients.

CORRECTIF

Servir les téléchargements par ID de document plutôt que par nom de fichier : charger l'enregistrement avec `WHERE id = ? AND customer_id = ?` et ne servir que le nom stocké. Vérifier en outre le chemin résolu par rapport au répertoire (`path.resolve` et `startsWith(uploadDir + path.sep)`) ou utiliser `res.download(name, { root: uploadDir })`.

SEC-08 · suite · Fichiers quelconques du serveur et documents d'autres clients téléchargeables

EXTRAIT

```
const file = String(req.query.file || '');
const fullPath = path.join(config.uploadDir, file);
...
res.download(fullPath);
```

SEC-09

La recherche de commandes permet de lire toute la base de données

MOYEN

CVSS 6,5 `AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N` · OWASP A05:2025 Injection · CWE-89 · Injection · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/orders.ts:22`
`src/routes/orders.ts:23`
`src/routes/orders.ts:24`

CONSTAT

`GET /api/orders/search` construit la requête SQL par concaténation des paramètres `q` et `status`. Le filtre client figure bien dans la requête, mais un `UNION SELECT` injecté permet de le contourner. La gestion centrale des erreurs renvoie les erreurs de base de données en clair à l'appelant, ce qui facilite les essais.

RISQUE

Un utilisateur connecté lit via `status` ou `q` n'importe quelle table, par ex. tous les utilisateurs avec e-mail, rôle et empreinte de mot de passe, ainsi que commandes et factures de tous les clients. L'écriture est impossible ici, la bibliothèque n'exécutant qu'une instruction de lecture ; les empreintes extraites permettent toutefois de prendre le contrôle de comptes.

CORRECTIF

Utiliser des paramètres liés : `... WHERE customer_id = ? AND status = ? AND reference LIKE ?` avec `.all(customerId, status, '%' + q + '%')`. Vérifier en outre `status` par rapport à la liste des valeurs autorisées.

EXTRAIT

```
const sql =
  `SELECT * FROM orders WHERE customer_id = ${req.user!.customerId} ` +
  `AND status = '${status}' AND reference LIKE '%${q}%'`;
res.json(db.prepare(sql).all());
```

SEC-10

Les factures d'autres clients sont consultables par leur ID

MOYEN

CVSS 6,5 `AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N` · OWASP A01:2025 Broken Access Control · CWE-639 · Autorisation · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/invoices.ts:20`
`src/routes/invoices.ts:24`
`src/routes/invoices.ts:28`
`src/services/cache.ts:5`

SEC-10 · suite · Les factures d'autres clients sont consultables par leur ID

CONSTAT `GET /api/invoices/:id` charge la facture par son seul ID, sans filtre sur le `customer_id` de l'utilisateur connecté. En outre, la facture reste dix minutes dans `invoiceCache`, dont la clé n'est que l'ID de facture ; une entrée en cache est servie sans aucun contrôle. Les autres routes de factures (`/`, `/:id/pdf`, `/:id/paid`) filtrent en revanche correctement par client.

RISQUE Un utilisateur connecté de n'importe quel client incrémente les ID de facture séquentiels et lit numéro de facture, montant, statut de paiement et commande associée de tous les autres clients.

CORRECTIF Passer la requête à `WHERE id = ? AND customer_id = ?`, comme dans `/:id/pdf`. Supprimer `invoiceCache` (voir le constat sur le cache des factures) ; s'il doit rester, clé `${customerId}:${id}` et invalidation sur `POST /:id/paid` ; mieux, centraliser le contrôle d'appartenance dans une fonction `getInvoiceForCustomer`.

EXTRAIT

```
const cached = invoiceCache.get(id);
if (cached) { return res.json(cached); }
const invoice = db.prepare('SELECT * FROM invoices WHERE id = ?').get(id);
```

SEC-11

L'inscription envoie un nombre illimité de courriels à n'importe quelle adresse

MOYEN

CVSS 6,5 `AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L` · OWASP A06:2025 Insecure Design · CWE-799 · Protection contre les abus · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT `src/routes/auth.ts:9`
`src/routes/auth.ts:21`

CONSTAT Chaque inscription anonyme déclenche immédiatement un courriel de bienvenue à l'adresse indiquée (`auth.ts:21`), sans limitation de débit, CAPTCHA ni confirmation de l'adresse.

RISQUE Un attaquant fait envoyer en masse, via le portail, des courriels à des adresses tierces et remplit la table des utilisateurs. Cela nuit à la réputation du domaine expéditeur (listes de blocage), consomme le quota d'envoi et peut bloquer les envois normaux.

CORRECTIF Limiter les inscriptions par IP et par fenêtre de temps, et n'activer les comptes qu'après confirmation de l'adresse ; avec l'invitation proposée dans le constat sur l'auto-inscription, l'endpoint ouvert disparaît totalement. Limiter de même la réinitialisation du mot de passe.

SEC-12

Le test de webhook appelle n'importe quelle adresse et affiche la réponse

MOYEN

CVSS 6,3 `AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:N/A:N` · OWASP A01:2025 Broken Access Control · CWE-918 · Appels externes · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/webhooks.ts:26`
`src/routes/webhooks.ts:28`
`src/routes/webhooks.ts:29`

CONSTAT

`POST /api/webhooks/test` appelle depuis le serveur, via `axios.get` et sans contrôle, l'URL issue du corps de la requête, puis renvoie le code de statut et l'intégralité de la réponse. Les adresses internes (`localhost`, réseaux privés, link-local) ne sont pas exclues, les redirections sont suivies. `POST /api/webhooks` enregistre lui aussi les URL sans contrôle.

RISQUE

Un utilisateur connecté se sert du serveur comme proxy vers le réseau interne et lit des services accessibles uniquement depuis celui-ci (par ex. interfaces d'administration ou, dans le cloud, le service de métadonnées et ses identifiants). La gravité dépend du réseau derrière le serveur, hors du dépôt ; d'où le classement en exploitation plus difficile.

CORRECTIF

Valider les URL à la création et avant chaque appel : `https` uniquement, résoudre les noms d'hôte et rejeter les adresses privées, de bouclage et link-local (par ex. avec une bibliothèque comme `ssrf-req-filter` en agent), désactiver les redirections (`maxRedirects: 0`). Dans le test, ne renvoyer que le code de statut et le temps de réponse, pas le contenu.

EXTRAIT

```
const { url } = req.body;
const response = await axios.get(url, { timeout: 5000 });
res.json({ status: response.status, body: response.data });
```

SEC-13

Connexions SMTP non chiffrées ou usurpées acceptées pour l'envoi des courriels

MOYEN

CVSS 5,9 `AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N` · OWASP A04:2025 Cryptographic Failures · CWE-295 · Chiffrement du transport · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/services/mail.ts:7`
`src/services/mail.ts:9`

CONSTAT

Le transport SMTP désactive la vérification des certificats avec `tls: { rejectUnauthorized: false }`. De plus, avec `secure: false` sur le port 587, STARTTLS n'est pas imposé (`requireTLS` manque) : la connexion s'établit aussi en clair si le serveur ou un attaquant ne propose pas STARTTLS.

RISQUE

Quiconque peut s'interposer entre le serveur du portail et le serveur SMTP (par ex. par manipulation DNS ou via un segment réseau compromis) intercepte les identifiants SMTP et peut ensuite envoyer des courriels au nom de la Muster AG ; contenus et destinataires des courriels sont également exposés.

SEC-13 · suite · Connexions SMTP non chiffrées ou usurpées acceptées pour l'envoi des courriels

CORRECTIF

Supprimer `rejectUnauthorized: false` et définir `requireTLS: true` (ou `secure: true` sur le port 465). Pour un serveur de messagerie interne doté de son propre certificat, déclarer son autorité de certification via `tls.ca` au lieu de désactiver la vérification.

EXTRAIT

```
secure: false,
auth: { user: config.smtp.user, pass: config.smtp.pass },
tls: { rejectUnauthorized: false },
```

SEC-14

Requêtes et téléversements très volumineux capables de paralyser le service

MOYEN

CVSS 5,9 `AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H` · OWASP A02:2025 Security Misconfiguration · CWE-770 · Limitation des ressources · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/server.ts:15`
`src/routes/auth.ts:29`
`src/routes/documents.ts:13`

CONSTAT

L'analyseur JSON accepte globalement jusqu'à 50 Mo par requête, y compris sur des endpoints anonymes comme `POST /api/auth/login`, qui écrit en plus tout le corps dans le journal. Le téléversement (`multer`) n'a aucune limite de taille ni de nombre de fichiers. Le dépôt ne montre pas si nginx limite la taille des requêtes (`client_max_body_size`) (hypothèse : non, ou largement).

RISQUE

Un attaquant anonyme peut épuiser mémoire et CPU de l'unique processus Node avec quelques requêtes parallèles de 50 Mo et rendre le portail inaccessible à tous les clients ; les utilisateurs connectés peuvent en outre remplir le disque par des téléversements. Une configuration restrictive de nginx supprime l'essentiel du risque.

CORRECTIF

Fixer globalement une limite JSON réaliste (par ex. `express.json({ limit: '100kb' })`), définir pour `multer` des `limits: { fileSize, files }` et limiter aussi la taille dans nginx. Ne plus journaliser le corps des requêtes.

EXTRAIT

```
app.use(express.json({ limit: '50mb' }));
```

SEC-15

Les fichiers téléversés écrasent les documents d'autres clients, sans limite de taille

MOYEN

CVSS 5,4 `AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:L` · OWASP A06:2025 Insecure Design · CWE-434 · Entrées et fichiers · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/documents.ts:10`
`src/routes/documents.ts:11`
`src/routes/documents.ts:13`

SEC-15 · suite · Les fichiers téléversés écrasent les documents d'autres clients, sans limite de taille

CONSTAT Tous les clients enregistrent dans le même répertoire, et le nom du fichier sur le serveur est le nom d'origine transmis par le navigateur. `multer` est configuré sans `limits` ni `fileFilter` ; taille, nombre et type de fichiers ne sont donc pas limités. Sortir du répertoire au téléversement est impossible, l'analyseur retirant les éléments de chemin du nom.

RISQUE Un utilisateur connecté téléverse un fichier portant le nom d'un document d'un autre client (par ex. `Vertrag.pdf`) et le remplace ; l'autre client reçoit au téléchargement le fichier substitué. Des fichiers volumineux ou nombreux peuvent remplir l'espace disque du serveur.

CORRECTIF Stocker les fichiers sous un nom généré côté serveur (par ex. UUID) dans un sous-répertoire par client et ne conserver le nom d'origine qu'en base. Définir dans `multer` des `limits` (`fileSize`, `files`) et un `fileFilter` pour les types autorisés.

EXTRAIT

```
destination: (_req, _file, cb) ⇒ cb(null, config.uploadDir),
filename: (_req, file, cb) ⇒ cb(null, file.originalname),
...
const upload = multer({ storage });
```

SEC-16

Erreurs renvoyées aux clients avec piles d'appels et détails internes

MOYEN

CVSS 5,3 `AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N` · OWASP A10:2025 Mishandling of Exceptional Conditions · CWE-209 · Gestion des erreurs · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT
`src/middleware/error.ts:5`
`src/middleware/error.ts:7`
`src/server.ts:26`

CONSTAT Le gestionnaire d'erreurs central renvoie pour chaque erreur `err.message` et l'intégralité de `err.stack` en JSON, quel que soit `NODE_ENV`. Il couvre toutes les routes, anonymes comprises : un simple JSON invalide envoyé à `POST /api/auth/login` déclenche une erreur d'analyse renvoyée avec la pile d'appels ; de même pour les erreurs de base (par ex. de `POST /api/auth/register` ou de la recherche de commandes).

RISQUE Un attaquant anonyme obtient sans effort chemins d'installation, bibliothèques et versions et – pour les erreurs de base – noms de tables et de colonnes ainsi que les messages d'erreur SQL. Cela facilite des attaques ciblées, notamment l'exploitation de l'injection SQL de la recherche de commandes via les messages d'erreur.

CORRECTIF Dans `errorHandler`, ne renvoyer qu'un message générique et un ID d'erreur, et ne journaliser les détails que côté serveur. Reprendre le code de statut de l'erreur (`err.status`, par ex. 400/413 pour les erreurs d'analyse) au lieu d'un 500 systématique.

EXTRAIT

```
res.status(500).json({
  error: err.message,
  stack: err.stack,
  path: req.path,
});
```

SEC-17

Mots de passe stockés en MD5 sans sel

MOYEN

CVSS 5,3 `AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:N/A:N` · OWASP A04:2025 Cryptographic Failures · CWE-916 · Cryptographie · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/auth/password.ts:5`
`src/auth/password.ts:9`
`src/routes/admin.ts:10`

CONSTAT

`hashPassword` calcule une simple empreinte MD5 sans sel (`password.ts:5`), et la comparaison n'est pas à temps constant (`:9`). L'endpoint d'administration `/api/admin/users` renvoie aussi les empreintes via `SELECT *` (`admin.ts:10`).

RISQUE

Quiconque obtient les empreintes – via l'endpoint d'administration, une sauvegarde ou une injection – retrouve en peu de temps la plupart des mots de passe par table de correspondance ou carte graphique, et peut les essayer sur d'autres services des utilisateurs.

CORRECTIF

Passer à Argon2id ou bcrypt. Migrer les comptes existants sans réinitialisation forcée : envelopper une fois l'empreinte MD5 dans Argon2id, puis la remplacer par une empreinte directe à la connexion suivante. Ne jamais renvoyer d'empreintes via l'API.

EXTRAIT

```
return crypto.createHash('md5').update(password).digest('hex');
```

SEC-18

Recensement des adresses e-mail enregistrées possible depuis l'extérieur

MOYEN

CVSS 5,3 `AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N` · OWASP A07:2025 Authentication Failures · CWE-204 · Authentification · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/auth.ts:33`
`src/routes/auth.ts:36`
`src/routes/auth.ts:51`
`src/middleware/error.ts:6`

CONSTAT

La connexion répond « Adresse e-mail inconnue » ou « Mot de passe incorrect » (`auth.ts:33` , `:36`), la réinitialisation renvoie 404 pour une adresse inconnue (`:51`). À l'inscription d'une adresse existante, la gestion des erreurs renvoie le message d'unicité de la base de données (`error.ts:6`).

RISQUE

Un attaquant anonyme teste des listes d'adresses et apprend qui est client de la Muster AG. Cela facilite l'hameçonnage ciblé et les attaques sur la connexion et la réinitialisation du mot de passe.

CORRECTIF

Toujours répondre à la connexion par le même message (« E-mail ou mot de passe incorrect »), à la réinitialisation toujours par 200 et un texte neutre. Ne renvoyer aucun message de base de données à l'inscription ; avec la confirmation par e-mail, seul le titulaire apprend que l'adresse existe déjà.

SEC-19

Vulnérabilités connues dans d'autres bibliothèques d'exécution

MOYEN

CVSS 5,3 `AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L` · OWASP A03:2025 Software Supply Chain Failures · CWE-1395 · Dépendances · Source : osv-scanner · ↑ Évaluation par domaine

EMPLACEMENT

`package.json:14`
`package.json:19`
`package.json:22`

CONSTAT

nodemailer 6.4.11, axios 0.21.1 et lodash 4.17.15 cumulent plus de cinquante avis publiés. Sont surtout exploitables les attaques par surcharge de l'analyseur d'adresses de nodemailer (l'adresse e-mail de l'inscription part sans contrôle à `sendMail`) et les faiblesses de proxy et de redirection d'axios dans le test de webhook. L'injection de commandes de nodemailer (CVE-2020-7769) ne concerne que le transport Sendmail, non utilisé ici.

RISQUE

Un attaquant anonyme peut ralentir sensiblement le processus via des adresses forgées à l'inscription ; les faiblesses d'axios aggravent la faille du webhook.

CORRECTIF

Mettre à jour nodemailer en ≥ 10.1.4, axios en ≥ 1.18.0 et lodash en 4.18 ou plus récent ; activer Dependabot ou Renovate pour les paquets d'exécution.

SEC-20

Les mots de passe en clair sont journalisés à chaque connexion

MOYEN

CVSS 5,1 `AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:L/A:N` · OWASP A09:2025 Security Logging and Alerting Failures · CWE-532 · Journalisation · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/auth.ts:29`

CONSTAT

La connexion écrit l'intégralité du corps de la requête dans le journal via `console.log` (`auth.ts:29`), donc l'adresse e-mail et le mot de passe en clair – y compris lors des échecs, qui contiennent souvent des mots de passe d'autres services.

RISQUE

Quiconque accède aux journaux du serveur, aux fichiers de log de pm2 ou à un service de journalisation connecté obtient les identifiants valides de tous les utilisateurs qui se sont connectés, administrateurs compris.

CORRECTIF

Supprimer la ligne et ne journaliser qu'un événement structuré sans mot de passe (résultat, ID utilisateur, IP). Effacer les journaux existants et inviter les utilisateurs concernés à changer de mot de passe.

EXTRAIT

```
console.log('Login-Versuch', req.body);
```

SEC-21

Les identifiants de la première version restent lisibles dans l'historique Git

MOYEN

CVSS 4,9 `AV:N/AC:H/PR:L/UI:N/S:C/C:L/I:L/A:N` · OWASP A02:2025 Security Misconfiguration · CWE-540 · Secrets · Source : gitleaks · ↑ Évaluation par domaine

EMPLACEMENT

```
.env:2 (Commit 8ba4686)
.env:4 (Commit 8ba4686)
.env:5 (Commit 8ba4686)
.env:6 (Commit 8ba4686)
```

CONSTAT

Le premier commit (8ba4686, 12/03/2026) contenait un `.env` avec la clé JWT, l'utilisateur et le mot de passe SMTP et la clé d'API d'un service de géolocalisation. Supprimé au commit suivant, le fichier reste lisible dans l'historique ; la recherche automatique ne signale que la clé d'API, nous avons trouvé les autres valeurs lors de la revue.

RISQUE

Quiconque a un accès en lecture au dépôt ou à un clone peut envoyer des courriels au nom du portail et utiliser le service de géolocalisation aux frais de la Muster AG. La validité des valeurs n'est pas vérifiable de l'extérieur ; la clé JWT est évaluée dans le constat sur la clé de repli.

CORRECTIF

Révoquer et réémettre le mot de passe SMTP et la clé du service de géolocalisation chez chaque fournisseur, régénérer la clé JWT comme indiqué dans le constat sur la clé de repli – supprimer le fichier ne suffit pas. Nettoyer ensuite l'historique avec `git filter-repo` et intégrer une recherche de secrets en hook pre-commit et dans la CI.

SEC-22

Jetons d'authentification valables sans limite et impossibles à révoquer

MOYEN

CVSS 4,8 `AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N` · OWASP A07:2025 Authentication Failures · CWE-613 · Authentification · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

```
src/auth/jwt.ts:13
src/auth/jwt.ts:20
src/middleware/auth.ts:20
src/middleware/auth.ts:28
```

CONSTAT

Une expiration n'est définie que si `JWT_EXPIRES_IN` est configuré (`jwt.ts:13`) ; sinon, un jeton reste valable indéfiniment. Il n'existe ni déconnexion ni liste de révocation, et rôle et rattachement client proviennent uniquement du jeton (`middleware/auth.ts:20` , `:28`), pas de la base de données.

RISQUE

Un jeton divulgué reste utilisable durablement. Le retrait d'un droit d'administrateur ou la suppression d'un utilisateur ne prend effet qu'à l'expiration du jeton – jamais si la variable n'est pas définie.

CORRECTIF

Fixer l'expiration dans le code (par ex. 15 à 60 minutes avec jeton de rafraîchissement) et définir `algorithms: ['HS256']` et `maxAge` dans `verifyToken` . Pour la révocation, stocker une version de jeton par utilisateur et la vérifier dans `requireAuth` avec le rôle et le statut issus de la base.

SEC-23

Signature des jetons vérifiée par une bibliothèque aux failles connues

MOYEN

CVSS 4,8 `AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N` · OWASP A03:2025 Software Supply Chain Failures · CWE-347 · Dépendances · Source : osv-scanner · ↑ Évaluation par domaine

EMPLACEMENT

`package.json:18`
`src/auth/jwt.ts:20`

CONSTAT

jsonwebtoken 8.5.1 est concerné par CVE-2022-23539, CVE-2022-23540 et CVE-2022-23541. `verifyToken` appelle `jwt.verify` sans liste fixe d'algorithmes. Comme un secret est toujours défini, le contournement connu n'est pas directement exploitable à ce jour.

RISQUE

Si la configuration des clés change (par ex. secret vide ou passage à des paires de clés), un attaquant peut faire accepter des jetons qu'il a lui-même signés et se connecter sous l'identité de n'importe quel utilisateur.

CORRECTIF

Mettre à jour jsonwebtoken en 9.0.x et fixer `algorithms: ['HS256']` dans `verifyToken`.

SEC-24

Le conteneur s'exécuterait en root avec tout le répertoire de travail

FAIBLE

CVSS 3,9 `AV:L/AC:H/PR:H/UI:N/S:U/C:L/I:L/A:L` · OWASP A02:2025 Security Misconfiguration · CWE-250 · Conteneur Docker · Source : semgrep, trivy · ↑ Évaluation par domaine

EMPLACEMENT

`Dockerfile:1`
`Dockerfile:4`
`Dockerfile:5`
`Dockerfile:9`

CONSTAT

Le Dockerfile ne définit pas de `USER`, l'application tourne donc en root. Sans `.dockerignore`, `COPY . .` reprend tout le contexte de build, lors d'un build local donc aussi un `.env`, `data/` et `uploads/` présents, dans une couche de l'image ; `npm install` y ajoute les dépendances de développement. D'après le pipeline, la production passe par rsync/pm2 ; le Dockerfile n'y semble pas utilisé.

RISQUE

Si l'image est utilisée, un attaquant capable d'exécuter du code dans le conteneur, ou de lire et d'écrire des fichiers via les failles de chemin du téléversement et du téléchargement, obtient les droits root dans le conteneur et donc l'accès à tous les fichiers, code compris. Les secrets d'un `.env` copié seraient lisibles par quiconque accède à l'image.

CORRECTIF

Build multi-étapes avec `npm ci` à l'étape de build et `npm ci --omit=dev` dans l'image d'exécution, puis `USER node` ; créer un `.dockerignore` avec `.env`, `data/`, `uploads/`, `.git`. Passer l'image de base à une version de Node maintenue, fixée par version ou digest, et ajouter un `HEALTHCHECK`. Si le Dockerfile n'est pas utilisé, le supprimer.

EXTRAIT

```
FROM node:16
WORKDIR /app
COPY . .
RUN npm install && npm run build
ENV NODE_ENV=production
CMD ["npm", "start"]
```

SEC-25

La connexion de déploiement ne vérifie pas l'identité du serveur de production

FAIBLE

CVSS 3,7 `AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N` · OWASP A08:2025 Software or Data Integrity Failures · CWE-295 · CI/CD · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`.github/workflows/deploy.yml:21`
`.github/workflows/deploy.yml:22`

CONSTAT

`rsync` et `ssh` s'exécutent avec `-o StrictHostKeyChecking=no` ; la clé d'hôte du serveur n'est donc jamais vérifiée. La clé privée de déploiement est en outre déposée sous forme de fichier `key` dans le répertoire de travail et n'est pas supprimée.

RISQUE

Si un attaquant peut détourner la connexion du runner (par ex. via le DNS), son serveur se fait passer pour le serveur de production et reçoit le code compilé de l'application ; le déploiement n'atteint alors pas le vrai serveur. Il n'obtient pas pour autant la clé privée.

CORRECTIF

Enregistrer la clé d'hôte du serveur comme secret ou variable et l'écrire dans `~/.ssh/known_hosts` avant le déploiement, supprimer `StrictHostKeyChecking=no`. Charger la clé via `ssh-agent` au lieu de la déposer en fichier dans l'espace de travail.

EXTRAIT

```
echo "${{ secrets.DEPLOY_KEY }}" > key
chmod 600 key
rsync -e "ssh -i key -o StrictHostKeyChecking=no" ...
```

SEC-26

Le pipeline de déploiement utilise des versions d'actions modifiables et des droits illimités

FAIBLE

CVSS 3,3 `AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:L/A:N` · OWASP A03:2025 Software Supply Chain Failures · CWE-1357 · CI/CD · Source : semgrep · ↑ Évaluation par domaine

EMPLACEMENT

`.github/workflows/deploy.yml:8`
`.github/workflows/deploy.yml:11`
`.github/workflows/deploy.yml:12`

CONSTAT

`actions/checkout@v3` et `actions/setup-node@v3` sont référencées par des tags déplaçables et non par des hachés de commit ; le workflow ne définit pas de bloc `permissions:`, les droits du `GITHUB_TOKEN` dépendent donc du réglage du dépôt (hypothèse : par défaut). Chaque push sur `main` part en production sans étape de contrôle (tests, validation via `environment`) ; une protection de branche n'est pas visible.

RISQUE

Si le tag d'une action intégrée est déplacé, du code tiers s'exécute dans le job même où la clé de déploiement se trouve sur le runner. Le risque est faible pour les actions maintenues par GitHub, suppose le contrôle de leur dépôt et relève donc du durcissement.

CORRECTIF

Épingler les actions sur le SHA de commit complet (mises à jour via Dependabot), définir `permissions: contents: read` et placer l'étape de déploiement dans un `environment` GitHub avec validation et job de test préalable. Garantir une protection de branche pour `main`.

SEC-26 · suite · Le pipeline de déploiement utilise des versions d'actions modifiables et des droits illimités

EXTRAIT

```
- uses: actions/checkout@v3
- uses: actions/setup-node@v3
```

SEC-27

Jetons d'authentification également acceptés dans l'URL

FAIBLE

CVSS 3,1 `AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N` · OWASP A07:2025 Authentication Failures · CWE-598 · Authentification · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT `src/middleware/auth.ts:15`

CONSTAT `requireAuth` accepte le jeton non seulement dans l'en-tête `Authorization`, mais aussi comme paramètre de requête `?token=` (`middleware/auth.ts:15`).

RISQUE Les URL aboutissent dans les journaux d'accès de nginx et des proxys, dans l'historique du navigateur et, via le `Referer`, chez des tiers. Quiconque y trouve un jeton agit au nom de l'utilisateur, durablement en l'absence d'expiration.

CORRECTIF Ne lire le jeton que dans l'en-tête `Authorization`. Pour les téléchargements qui doivent se passer d'en-tête, émettre des liens de téléchargement de courte durée et à usage unique.

EXTRAIT

```
const token = header.startsWith('Bearer ') ? header.slice(7) : req.query.token;
```

SEC-28

La configuration CORS par défaut autorise l'accès depuis n'importe quel site

FAIBLE

CVSS 3,1 `AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N` · OWASP A02:2025 Security Misconfiguration · CWE-942 · CORS · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT `src/server.ts:14`
`src/config.ts:9`

CONSTAT Si `CORS_ORIGIN` n'est pas défini, la configuration CORS (`origin: true`) renvoie toute origine demandeuse en autorisant l'envoi d'identifiants. L'authentification passant aujourd'hui uniquement par un jeton Bearer dans l'en-tête ou le paramètre de requête, sans cookies, une page tierce ne peut pas récupérer de données au nom d'un utilisateur. On ignore si `CORS_ORIGIN` est défini en production (hypothèse ; `.env.example` le prévoit).

RISQUE Pas de chemin d'attaque immédiat. Dès que des cookies ou d'autres identifiants envoyés automatiquement seront introduits, tout site visité par un utilisateur connecté pourrait lire ses commandes et factures.

CORRECTIF Sans variable définie, ne pas se rabattre sur `true`, mais interrompre le démarrage ou utiliser une liste fixe d'origines autorisées (`origin: ['https://portal.muster-ag.example']`) ; ne définir `credentials: true` qu'en cas de besoin réel.

SEC-28 · suite · La configuration CORS par défaut autorise l'accès depuis n'importe quel site

EXTRAIT

```
app.use(cors({ origin: config.corsOrigin || true, credentials: true }));
```

SEC-29

Les en-têtes de sécurité manquent, le serveur révèle technologie et version

FAIBLE

CVSS 3,1 `AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N` · OWASP A02:2025 Security Misconfiguration · CWE-693 · En-têtes de sécurité · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/server.ts:12`
`src/server.ts:17`

CONSTAT

L'application ne définit aucun en-tête de sécurité (ni `helmet`, ni HSTS, ni `X-Content-Type-Options`, ni options de frame) et laisse actif l'en-tête par défaut `X-Powered-By: Express` ; `GET /health` renvoie anonymement la version de l'application. Le dépôt ne montre pas si nginx ajoute ces en-têtes (hypothèse).

RISQUE

Pris isolément, pas de chemin d'attaque direct, l'API renvoyant du JSON et les fichiers en téléchargement. L'absence de HSTS facilite toutefois les attaques par rétrogradation de la connexion du navigateur, et les indications de technologie et de version aident à choisir des exploits adaptés.

CORRECTIF

Intégrer `helmet()` ou définir `app.disable('x-powered-by')`, garantir HSTS et `X-Content-Type-Options: nosniff` dans l'application ou dans nginx, et retirer la version de `/health`.

EXTRAIT

```
app.get('/health', (_req, res) => res.json({ ok: true, version: process.env.npm_package_version
}));
```

SEC-30

La liste des utilisateurs de l'administration renvoie les empreintes de mots de passe

FAIBLE

CVSS 2,7 `AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N` · OWASP A01:2025 Broken Access Control · CWE-200 · Autorisation · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/admin.ts:10`

CONSTAT

`GET /api/admin/users` renvoie `SELECT * FROM users`, donc aussi la colonne `password_hash`. L'empreinte n'est pas nécessaire à la gestion des utilisateurs.

RISQUE

Quiconque dispose de droits d'administrateur ou obtient un jeton administrateur récupère les empreintes de tous les comptes et peut les casser hors ligne ; avec l'inscription ouverte en tant qu'administrateur, c'est possible sans connaissance préalable.

CORRECTIF

Ne renvoyer que les colonnes nécessaires (`SELECT id, customer_id, email, role, created_at FROM users`) ; de manière générale, éviter `SELECT *` dans les réponses de l'API.

SEC-30 · suite · La liste des utilisateurs de l'administration renvoie les empreintes de mots de passe

EXTRAIT

```
res.json(db.prepare('SELECT * FROM users').all());
```

CHAPITRE 4

Qualité, exploitation et dépendances

Chaque domaine est évalué sur quatre niveaux, ses constats suivent directement. Les constats qualité sont hiérarchisés, mais non cotés selon CVSS.

4.1 Qualité du code et maintenabilité

■■□□ Actions requises

Le SQL figure directement dans les handlers de requête, et les commandes disposent de deux voies d'accès parallèles. Le contrôle client doit donc être réécrit à la main à chaque endroit – c'est précisément là que naissent les failles de ce rapport. Les erreurs d'envoi de courriels sont ignorées ou laissent des états incomplets. Le mode strict de TypeScript est actif, mais largement contourné par des lignes de base non typées.

QA-06

Deux styles d'accès aux données, avec du SQL directement dans les routes

■■□□ MOYEN

Architecture · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/orders.ts:12`
`src/routes/orders.ts:31`
`src/routes/invoices.ts:24`
`src/routes/admin.ts:10`
`src/db.ts:10`

CONSTAT

Tous les routeurs contiennent du SQL directement dans le handler de requête ; il n'existe pas de couche métier. En parallèle, deux endpoints de commandes utilisent un repository générique et les autres endpoints du même fichier du SQL brut – `orders.ts` a donc deux façons de lire les commandes. Le schéma est créé au démarrage par `CREATE TABLE IF NOT EXISTS` ; il n'existe aucune migration pour les évolutions futures.

RISQUE

Des règles comme le contrôle client doivent être réécrites à la main dans chaque requête et manquent donc par endroits ; une modification de table touche des endroits dispersés. Sans migrations, modifier les colonnes d'une base de production existante n'est possible qu'à la main.

CORRECTIF

Créer par domaine métier un module de données léger avec des fonctions nommées (par ex. `findOrdersForCustomer(customerId)`, `findInvoice(id, customerId)`), où le filtre client est un paramètre obligatoire ; les routeurs n'appellent plus que ces fonctions. Introduire un outil de migration simple (des fichiers SQL numérotés et le `user_version` de SQLite suffisent).

QA-06 · suite · Deux styles d'accès aux données, avec du SQL directement dans les routes

EXTRAIT

```
const orders = db
  .prepare('SELECT * FROM orders WHERE customer_id = ? ORDER BY created_at DESC')
  .all(customerId) as any[];
...
const order = repositories.get('order').findById(Number(req.params.id));
```

QA-07

Entrées jamais validées à l'interface de l'API

MOYEN

OWASP A06:2025 Insecure Design · CWE-20 · Validation des entrées · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/server.ts:15`
`src/routes/auth.ts:11`
`src/routes/webhooks.ts:14`
`src/routes/admin.ts:19`

CONSTAT

Aucune route ne valide le corps ou les paramètres de la requête par un schéma ; les valeurs sont déstructurées directement et transmises à la base, à l'envoi de courriels ou au client HTTP. L'analyseur JSON accepte des corps jusqu'à 50 Mo, y compris sur les routes anonymes. `POST /api/admin/settings` fusionne le corps avec des valeurs par défaut via `_.merge`, sans enregistrer le résultat.

RISQUE

L'absence de contrôle des types et des champs est la cause commune des constats sur l'inscription, les webhooks et la recherche, et favorise de nouvelles failles à chaque évolution. Les gros corps JSON bloquent le processus pendant l'analyse.

CORRECTIF

Imposer sur chaque route un schéma (par ex. zod ou express-validator) avec champs, types et longueurs autorisés, et rejeter les champs inconnus. Ramener la limite JSON à une valeur réaliste comme `100kb` ; supprimer la route sans effet `/api/admin/settings` ou la réaliser sans `_.merge` sur des données externes.

EXTRAIT

```
app.use(express.json({ limit: '50mb' }));
```

QA-08

Erreurs d'envoi de courriels perdues ou laissant des états incohérents

MOYEN

CWE-755 · Gestion des erreurs · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/core/EventBus.ts:19`
`src/core/EventBus.ts:30`
`src/routes/auth.ts:18`
`src/routes/auth.ts:21`
`src/routes/auth.ts:47`

QA-08 · suite · Erreurs d'envoi de courriels perdues ou laissant des états incohérents

CONSTAT Le bus d'événements ignore toute erreur de ses handlers avec `.catch(() => undefined)` : si le courriel d'annulation échoue, il n'y a ni log ni nouvel essai. À l'inscription, l'utilisateur est d'abord enregistré, puis le courriel de bienvenue attendu ; si l'envoi échoue, le client reçoit une erreur 500 alors que le compte existe. Le handler de réinitialisation est `async`, mais ne transmet pas les erreurs à `next` (voir le constat sur le plantage).

RISQUE Des clients ne reçoivent pas leurs confirmations d'annulation sans que personne ne le remarque ; en cas de panne de messagerie, des inscriptions restent dans un état incohérent qui ne se résout que par une intervention manuelle en base.

CORRECTIF Dans le bus d'événements, ou dans le futur appel direct, au minimum journaliser les erreurs (`console.error` ou un logger avec l'ID de commande). Découpler l'envoi de la requête d'inscription (envoyer après l'insertion, journaliser au lieu d'un 500) et brancher tous les handlers `async` sur `next` via un wrapper comme `asyncHandler(fn)`, ou passer à Express 5, qui transmet lui-même les promesses rejetées.

EXTRAIT

```
for (const handler of this.handlers[event] || []) {
  Promise.resolve(handler(payload)).catch(() => undefined);
}
```

QA-11

Événements de sécurité non journalisés

MOYEN

OWASP A09:2025 Security Logging and Alerting Failures · CWE-778 · Journalisation · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT
`src/routes/auth.ts:28`
`src/routes/auth.ts:47`
`src/routes/admin.ts:17`
`src/routes/admin.ts:23`

CONSTAT Connexions réussies ou échouées, réinitialisations de mot de passe, inscriptions avec rôle et actions d'administration comme l'export des utilisateurs (`admin.ts:23`) ne laissent aucune trace dans le journal. Le seul appel de journalisation de la connexion est la sortie en clair du corps de la requête.

RISQUE Les attaques sur des comptes ou une fuite de données via l'export passent inaperçues et ne peuvent pas être reconstituées après coup ; cela complique aussi une notification au titre de l'article 33 du RGPD.

CORRECTIF Mettre en place un journal d'audit structuré (par ex. avec `pino`) pour connexion, réinitialisation, inscription, changement de rôle, export et appels d'administration, avec horodatage, ID utilisateur, client et IP ; le conserver séparément et alerter en cas d'anomalie, par ex. des échecs répétés.

QA-12

Absence de typage des lignes de base et des entrées, le mode strict de TypeScript est contourné

FAIBLE

Lisibilité et typage · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/auth.ts:11`
`src/routes/auth.ts:31`
`src/routes/orders.ts:14`
`src/routes/invoices.ts:35`
`src/core/EventBus.ts:27`
et 2 autres emplacements du même type

CONSTAT

`tsconfig.json` active `strict`, mais les résultats de la base sont presque partout réutilisés avec `as any` ou `as any[]`, et les corps de requête sont déstructurés sans schéma. Le contrôle de types ne détecte ainsi ni les fautes de frappe dans les noms de colonnes, ni les champs manquants ou mal typés dans les requêtes.

RISQUE

Des erreurs comme un nom de colonne erroné ou une chaîne au lieu d'un nombre n'apparaissent qu'à l'exécution ; la sûreté promise par le mode strict n'existe pas en pratique.

CORRECTIF

Créer une interface de ligne par table (comme `Order` dans `OrderRepository.ts`) et typer les requêtes avec (`db.prepare<[number], Invoice>(...)`). Valider les corps de requête avec une bibliothèque de schémas comme `zod` et en dériver les types ; activer la règle ESLint `@typescript-eslint/no-explicit-any`.

EXTRAIT

```
const user = db.prepare('SELECT * FROM users WHERE email = ?').get(email) as any;
```

4.2 Complexité et volume

Bon

Aucune fonction n'atteint une complexité cyclomatique élevée, et il n'y a ni fonctions trop longues ni longues listes de paramètres. Les commandes disposent pourtant d'un repository générique avec fabrique et d'un bus d'événements maison, alors qu'il n'existe qu'un cas d'usage pour chacun. Les deux se remplacent sans perte fonctionnelle par quelques fonctions directes ; cela économise environ 100 des quelque 600 lignes de code source, soit environ un sixième.

INDICATEURS DE COMPLEXITÉ ET DE VOLUME

Mesurés avec lizard sur le code de production, hors tests, dépendances et code généré. La complexité cyclomatique (CCN) compte les chemins indépendants à travers une fonction.

INDICATEUR	SEUIL	TYPESCRIPT	TOTAL
Fonctions		48	48
Lignes de code (NLOC)		309	309
Complexité moyenne	≤ 5	1,9	1,9
Complexité 90e centile	≤ 10	3	3
Complexité maximale	≤ 20	7	7
Fonctions avec CCN > 10	< 5 %	0 0,0 %	0 0,0 %
Fonctions avec CCN > 20	< 1 %	0 0,0 %	0 0,0 %
Fonctions > 60 lignes	< 2 %	0 0,0 %	0 0,0 %
Fonctions > 5 paramètres	< 1 %	0 0,0 %	0 0,0 %
Code dupliqué	< 5 %		0,0 %
Part de test (lignes de test / total)	≥ 30 %		8,8 %

QA-09

Fabrique et Repository génériques pour deux requêtes simples

MOYEN

Complexité et volume · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/core/AbstractRepository.ts:12`
`src/core/AbstractRepository.ts:20`
`src/core/AbstractRepository.ts:41`
`src/core/OrderRepository.ts:11`
`src/core/RepositoryFactory.ts:8`
et 3 autres emplacements du même type

CONSTAT

Pour les commandes, il existe une classe de base abstraite, une seule classe dérivée et une fabrique avec registre et cache d'instances, qui ne connaît qu'un repository (`'order'`). Seuls `findById` et `update` sont utilisés, dans `orders.ts` ; `findAll` et `map` ne sont appelés nulle part. Contre-vérification : ni seconde implémentation, ni test qui remplace le repository, ni autre base – les autres tables sont lues directement en SQL.

QA-09 · suite · Fabrique et Repository génériques pour deux requêtes simples

RISQUE

La couche double les voies d'accès aux commandes et impose trois indirections pour une lecture par clé primaire. Le `findAll` inutilisé insère tri et limite sans contrôle dans le SQL et deviendrait une porte d'entrée s'il servait un jour avec des saisies utilisateur ; ce type de code mort est facilement réutilisé sans y penser.

CORRECTIF

Remplacer les trois fichiers par deux fonctions nommées, par ex. `findOrderForCustomer(id, customerId)` et `setOrderStatus(id, status)`, chacune avec une requête préparée – estimation 89 lignes → 10 à 15 lignes, soit environ 75 à 80 lignes en moins. Le filtre client va alors directement dans la requête plutôt que dans la comparaison du routeur.

EXTRAIT

```
type RepositoryName = 'order';
...
export const repositories = new RepositoryFactory().register('order', OrderRepository);
```

QA-13

Bus d'événements maison pour un seul événement et un seul destinataire

FAIBLE

Complexité et volume · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/core/EventBus.ts:6`
`src/core/EventBus.ts:10`
`src/core/EventBus.ts:26`
`src/routes/orders.ts:45`

CONSTAT

`EventBus.ts` implémente un mécanisme publish/subscribe typé (32 lignes) pour un seul événement (`order.cancelled`), qui n'a qu'un destinataire, enregistré dans le même fichier, et n'est déclenché qu'à un endroit. Contre-vérification : aucun autre événement, aucune livraison de webhooks via le bus (la table `webhooks` n'y est pas reliée), aucune distribution entre processus, aucun test.

RISQUE

Ce détour masque ce qui se passe réellement à l'annulation (envoi d'un courriel) et explique pourquoi les erreurs y sont ignorées en silence (voir le constat sur les erreurs perdues à l'envoi de courriels). Sans second cas d'usage, l'abstraction n'apporte rien.

CORRECTIF

Placer le destinataire dans une fonction `notifyOrderCancelled(orderId, customerId)` de `services/mail.ts` ou d'un module de commandes et l'appeler directement dans `orders.ts` (sans `await`, avec `.catch(logError)`) – estimation 32 lignes → 10 à 14 lignes, soit environ 18 à 22 lignes en moins. N'introduire un bus que si des webhooks ou d'autres destinataires arrivent réellement.

EXTRAIT

```
interface EventMap {
  'order.cancelled': { orderId: number; customerId: number };
}
```

QA-14

Endpoint de réglages sans effet, qui ajoute sa propre bibliothèque

FAIBLE

Complexité et volume · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/admin.ts:2`
`src/routes/admin.ts:17`
`package.json:19`

CONSTAT

`POST /api/admin/settings` fusionne valeurs par défaut et corps de requête avec `lodash.merge` et renvoie le résultat, sans rien enregistrer ; `invoicePrefix` et `reminderDays` ne sont lus nulle part. Lodash n'est intégré que pour cet appel. Le dépôt ne montre pas si un frontend utilise l'endpoint comme aperçu ; les vulnérabilités connues de lodash sont évaluées dans le constat sur les autres bibliothèques d'exécution.

RISQUE

Les administrateurs pensent avoir modifié des réglages qui restent sans effet ; la dépendance supplémentaire doit être maintenue sans utilité.

CORRECTIF

Supprimer l'endpoint et Lodash ou – si des réglages sont nécessaires – créer une table `settings` et y enregistrer les valeurs par décomposition d'objet (`{ ...defaults, ...validated }`) après validation par schéma. Gain : environ 5 à 6 lignes et une dépendance d'exécution.

EXTRAIT

```
adminRouter.post('/settings', (req, res) ⇒ {
  const defaults = { invoicePrefix: 'RE-', reminderDays: 14 };
  const settings = _.merge({}, defaults, req.body);
  res.json(settings);
});
```

4.3 Performance et cache

Actions requises

Le cache des factures est créé sans ID client dans la clé et n'est pas invalidé lors du marquage comme payée. La source, un accès SQLite local par clé primaire, est de toute façon rapide ; il doit donc disparaître. L'aperçu des commandes interroge les lignes une à une pour chaque commande, et les colonnes client n'ont pas d'index. Avec les volumes actuels, cela ne se remarque pas encore.

QA-04

Chaque commande de l'aperçu déclenche sa propre requête en base

MOYEN

Performance et cache · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/orders.ts:12`
`src/routes/orders.ts:15`
`src/routes/orders.ts:16`
`src/db.ts:32`

CONSTAT

`GET /api/orders` charge toutes les commandes d'un client, puis interroge en boucle les lignes de chaque commande une à une (schéma N+1) ; la requête est de plus préparée à nouveau à chaque itération. `order_items.order_id` n'a pas d'index, SQLite n'en crée pas automatiquement pour les clés étrangères – chacune de ces requêtes parcourt donc toute la table des lignes.

QA-04 · suite · Chaque commande de l'aperçu déclenche sa propre requête en base

RISQUE Le temps de réponse croît avec le nombre de commandes multiplié par le nombre de lignes ; pour les clients à long historique, l'aperçu ralentit sensiblement et, better-sqlite3 étant synchrone, bloque pendant ce temps toutes les autres requêtes du processus (estimation, non mesurée).

CORRECTIF Charger les lignes en une requête (`SELECT * FROM order_items WHERE order_id IN (SELECT id FROM orders WHERE customer_id = ?)`) et les grouper par `order_id` dans le code, préparer les requêtes une seule fois au niveau du module et ajouter `CREATE INDEX order_items_order_id ON order_items(order_id)` .

EXTRAIT

```
for (const order of orders) {
  order.items = db.prepare('SELECT * FROM order_items WHERE order_id = ?').all(order.id);
}
```

QA-10

Le cache des factures sert des paiements périmés et contourne tout contrôle client

MOYEN

CWE-524 · Performance et cache · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/services/cache.ts:5`
`src/config.ts:16`
`src/routes/invoices.ts:20`
`src/routes/invoices.ts:28`
`src/routes/invoices.ts:42`

CONSTAT Le seul cache de données (`invoiceCache` , en mémoire du processus, illimité) stocke des factures complètes sous leur ID, sans ID client dans la clé, et n'est pas invalidé lors du marquage comme payée. La source est une lecture par clé primaire dans un fichier SQLite local, déjà rapide. Le cache précédant la requête en base, corriger la seule requête SQL ne ferme pas la faille de SEC-10.

RISQUE Après `POST /api/invoices/:id/paid` , la vue détaillée affiche encore « non payée » jusqu'à dix minutes ; une entrée mise en cache est servie à tout utilisateur connecté, quel que soit son client.

CORRECTIF Supprimer le cache sans remplacement et lire la facture directement avec `WHERE id = ? AND customer_id = ?` ; `node-cache` et `invoiceCacheTtl` disparaissent. Si un cache devient nécessaire plus tard, inclure l'ID client dans la clé, invalider à chaque modification et définir `maxKeys` .

EXTRAIT

```
const cached = invoiceCache.get(id);
if (cached) {
  return res.json(cached);
}
const invoice = db.prepare('SELECT * FROM invoices WHERE id = ?').get(id);
```

QA-15

Index absents sur les colonnes client qui filtrent chaque liste

FAIBLE

Performance et cache · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/db.ts:26`
`src/db.ts:41`
`src/db.ts:50`
`src/db.ts:57`
`src/routes/invoices.ts:13`
et 1 autre emplacement du même type

CONSTAT

Hormis les clés primaires et `users.email`, le schéma ne crée aucun index. Toutes les listes client (commandes, factures, documents, webhooks) filtrent sur `customer_id` et doivent donc parcourir entièrement la table concernée.

RISQUE

Sans effet visible sur de petits volumes ; avec la croissance du nombre de clients, toutes les listes ralentissent uniformément, alors que chaque client ne voit qu'une petite partie des données (estimation, non mesurée).

CORRECTIF

Créer des index sur `customer_id` dans `orders`, `invoices`, `documents`, `webhooks` et `users`, pour `orders` combiné avec `created_at` (`(customer_id, created_at)`) selon le tri de l'aperçu – idéalement comme première migration.

EXTRAIT

```
CREATE TABLE IF NOT EXISTS invoices (
  id INTEGER PRIMARY KEY,
  customer_id INTEGER NOT NULL REFERENCES customers(id),
```

QA-16

Listes et export de l'administration chargent tous les enregistrements d'un coup

FAIBLE

Performance et cache · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`src/routes/orders.ts:13`
`src/routes/invoices.ts:13`
`src/routes/documents.ts:19`
`src/routes/admin.ts:10`
`src/routes/admin.ts:24`

CONSTAT

Aucune route de liste ne connaît la pagination ni de limite ; les routes d'administration `/users`, `/customers` et `/export` chargent chacune toute la table en mémoire avec `.all()` et la sérialisent en une seule réponse JSON.

RISQUE

Mémoire et temps de réponse croissent linéairement avec le volume de données ; un export de tous les utilisateurs bloque le processus synchrone pendant toute la durée de la requête (estimation, non mesurée).

CORRECTIF

Introduire pour les listes client `limit` / `offset` ou, mieux, une pagination par clé (`WHERE id < ? ORDER BY id DESC LIMIT 50`) ; produire l'export en flux CSV ou NDJSON avec `stmt.iterate()`.

QA-16 · suite · Listes et export de l'administration chargent tous les enregistrements d'un coup

EXTRAIT

```
adminRouter.get('/users', (_req, res) => {
  res.json(db.prepare('SELECT * FROM users').all());
});
```

4.4 Tests

■□□□ Insuffisant

Seules les fonctions de mot de passe et l'exigence d'authentification de la liste des commandes sont vérifiées. Factures, documents, webhooks, administration, inscription et réinitialisation du mot de passe n'ont pas de tests, et il n'existe pas de données de test avec plusieurs clients. La configuration Jest pour TypeScript manque, et les tests ne tournent dans aucun pipeline.

QA-02

Les processus critiques ne sont pratiquement pas couverts par des tests

■■■□ ÉLEVÉ

Tests · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`tests/orders.test.ts:7`
`tests/password.test.ts:3`
`package.json:11`
`tsconfig.json:11`

CONSTAT

Il existe deux fichiers de test : l'un vérifie que `GET /api/orders` exige une authentification, l'autre teste les fonctions de mot de passe. Inscription, réinitialisation, factures, documents, webhooks et administration n'ont aucun test, et aucun ne vérifie le cloisonnement des clients. La configuration Jest pour TypeScript manque (`ts-jest` est installé mais non déclaré), et le test des commandes s'exécute sur le vrai fichier de base.

RISQUE

Les erreurs de logique d'accès et de cloisonnement – relevées à plusieurs reprises dans cet audit – n'apparaissent qu'en production ; chaque modification d'une route se fait à l'aveugle. L'analyse statique ne garantit pas que les tests existants s'exécutent dans la configuration actuelle.

CORRECTIF

Configurer Jest avec `preset: 'ts-jest'` et exécuter les tests sur une base SQLite en mémoire avec deux clients de test (`DB_FILE=:memory:` plus jeu de données). Écrire d'abord des tests d'intégration du cloisonnement par route, du contrôle de rôle de l'administration, de la connexion et de la réinitialisation ; puis couvrir chaque correctif de ce rapport par un test de non-régression.

EXTRAIT

```
it('liefert die Bestellungen des eigenen Kunden', async () => {
  const res = await request(app).get('/api/orders').set('Authorization', `Bearer ${token}`);
  expect(res.status).toBe(200);
  expect(Array.isArray(res.body)).toBe(true);
});
```

4.5 Build, CI/CD et exploitation

Actions requises

Chaque push sur la branche principale est construit et livré aussitôt, sans tests, linter ni contrôle de pull request comme garde-fou. Node.js 16 ne reçoit plus de mises à jour, et le lockfile n'étant pas copié sur le serveur, celui-ci résout à nouveau les dépendances à chaque déploiement. Un Dockerfile inutilisé coexiste avec le circuit de déploiement réellement employé. Les points relevant de la sécurité sont évalués en SEC-07 et SEC-24 à SEC-26.

QA-01

Chaque push sur main part en production sans tests ni contrôles

ÉLEVÉ

Build et CI/CD · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`.github/workflows/deploy.yml:4`
`.github/workflows/deploy.yml:15`
`.github/workflows/deploy.yml:16`
`package.json:7`

CONSTAT

L'unique pipeline construit à chaque push sur `main` et copie le résultat directement sur le serveur de production. Les tests (`npm test`) ne sont pas lancés, aucun linter n'est configuré ni disponible en script, et il n'y a pas de pipeline pour les pull requests. Le seul garde-fou est le compilateur TypeScript lors du build.

RISQUE

Un commit défectueux atteint sans préavis le portail client public ; les régressions sur des erreurs déjà corrigées ne sont pas détectées. Le dépôt ne montre pas si une protection de branche ou des revues sont configurées côté GitHub (hypothèse : non).

CORRECTIF

Créer un job de contrôle dédié pour les pull requests et `main` (`npm ci`, `tsc --noEmit`, linter, `npm test`) et en faire dépendre le job de déploiement via `needs:`. Introduire ESLint avec `@typescript-eslint` et activer pour `main` une protection de branche avec contrôle obligatoire.

EXTRAIT

```
on:
  push:
    branches: [main]
...
      - run: npm install
      - run: npm run build
```

QA-03

Aucune maintenance des dépendances, la plupart en retard de versions majeures

MOYEN

Build et CI/CD · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`package.json:13`
`package.json:24`

QA-03 · suite · Aucune maintenance des dépendances, la plupart en retard de versions majeures

CONSTAT Les dépendances directes sont figées sur des versions de 2019 à 2022 (notamment Express 4.17.1, axios 0.21.1, jsonwebtoken 8.5.1, multer 1.4.2, nodemailer 6.4.11, better-sqlite3 7.6.2), parfois plusieurs versions majeures en retard. Aucun processus de mise à jour automatique (Dependabot, Renovate) n'est en place. Les vulnérabilités connues qui en découlent sont évaluées à part, comme constats de sécurité.

RISQUE Plus les mises à jour tardent, plus le saut devient important et risqué ; les correctifs de sécurité ne peuvent alors pas être appliqués rapidement, car il faut d'abord absorber les ruptures de compatibilité de plusieurs versions.

CORRECTIF Mettre en place Dependabot ou Renovate avec des mises à jour hebdomadaires groupées, fusionnées uniquement après le job de contrôle de la CI. Rattraper les sauts de versions majeures un par un, avec l'appui des tests, en commençant par les paquets à vulnérabilités connues.

EXTRAIT

```
"axios": "0.21.1",
"better-sqlite3": "7.6.2",
"express": "4.17.1",
"jsonwebtoken": "8.5.1",
```

QA-05

Des builds non reproductibles, sur une version de Node.js qui n'est plus maintenue

MOYEN

Build et CI/CD · Source : Revue · ↑ Évaluation par domaine

EMPLACEMENT

`.github/workflows/deploy.yml:14`
`.github/workflows/deploy.yml:15`
`.github/workflows/deploy.yml:21`
`.github/workflows/deploy.yml:22`
`Dockerfile:1`
et 1 autre emplacement du même type

CONSTAT La CI et le Dockerfile utilisent Node 16, sans mises à jour de sécurité depuis septembre 2023. Le pipeline utilise `npm install` au lieu de `npm ci`, et seuls `dist` et `package.json` partent sur le serveur, pas `package-lock.json` – les dépendances indirectes y sont résolues à nouveau à chaque déploiement. S'y ajoute un Dockerfile inutilisé sans `.dockerignore` (`COPY . .` reprend aussi un `.env` local).

RISQUE La production peut contenir d'autres versions de paquets que la version testée ; les erreurs deviennent alors impossibles à reproduire. Deux modes d'exploitation distincts et non alignés (pm2 via rsync et Docker) compliquent maintenance et diagnostic.

CORRECTIF Passer à une version LTS actuelle de Node et la fixer dans `.nvmrc` ou `engines` ; utiliser `npm ci` avec le lockfile livré, en CI comme sur le serveur. Choisir un seul mode d'exploitation – par ex. construire l'image Docker en CI (multi-étapes, avec `.dockerignore`, dépendances de production seules) et livrer exactement cet artefact.

QA-05 · suite · Des builds non reproductibles, sur une version de Node.js qui n'est plus maintenue

EXTRAIT

```
rsync ... -r dist package.json deploy@portal.muster-ag.example:/srv/portal
ssh ... "cd /srv/portal && npm install --production && pm2 restart portal"
```

4.6 Dépendances

■□□□ Insuffisant

Sont notamment concernés Express, jsonwebtoken, Multer, axios, lodash et nodemailer. Sans authentification, on atteint surtout un plantage via l'analyseur JSON (body-parser) et une attaque par surcharge via l'analyseur de requêtes d'Express ; les autres avis aggravent des failles existantes. Les licences ne posent pas de problème, le lockfile est complet, il n'y a aucune source de paquets tierce. Les constats individuels sont évalués en SEC-04, SEC-19 et SEC-23.

CHAPITRE 5

Analyse automatisée

Les outils détectent de manière fiable ce qui relève de motifs connus. Les résultats pertinents ont été vérifiés dans le code, les faux positifs écartés. Les résultats bruts sont joints.

OUTIL	CONTRÔLE	BRUTS	PERTINENTS	CONSTATS
gitleaks	Secrets dans tout l'historique Git	1	1	SEC-21
osv-scanner	Dépendances face aux CVE/OSV, licences	83	3	SEC-04, SEC-19, SEC-23
semgrep	SAST, jeux de règles OWASP Top 10 et Security Audit	6	4	SEC-08, SEC-24, SEC-26
trivy	Configuration : Dockerfiles, IaC	2	2	SEC-24
lizard	Complexité cyclomatique, longueur des fonctions, paramètres, duplications	48	0	–

ACTUALITÉ DU FRAMEWORK ET DE L'ENVIRONNEMENT D'EXÉCUTION

COMPOSANT	EN SERVICE	DERNIÈRE	APPRÉCIATION
Node.js (exécution)	16	26 LTS	sans mises à jour de sécurité depuis 2023
axios	0.21.1	1.21.0	1 version majeure de retard
better-sqlite3	7.6.2	13.0.3	6 versions majeures de retard

COMPOSANT	EN SERVICE	DERNIÈRE	APPRÉCIATION
express	`4.17.1`	`5.3.0`	1 version majeure de retard
jsonwebtoken	`8.5.1`	`9.0.3`	1 version majeure de retard
multer	`1.4.2`	`2.5.0`	1 version majeure de retard
nodemailer	`6.4.11`	`10.3.1`	4 versions majeures de retard

PROVENANCE ET ÉPINGLAGE DES DÉPENDANCES

- Lockfile complet, aucune source de paquets tierce, pas de .npmrc propre
- Scripts d'installation dans trois paquets (better-sqlite3, fsevents, nodemailer) – courant, mais exécutés sur le serveur sans lockfile
- GitHub Actions référencées par des tags modifiables, droits du jeton non limités

LICENCES

Aperçu sans appréciation juridique.

MIT	389	BSD-2-Clause	1
ISC	32	CC-BY-4.0	1
BSD-3-Clause	18	CC0-1.0 OR MIT	1
Apache-2.0	7	MIT OR WTFPL	1
Apache-2.0 OR BSD-2-Clause OR MIT	1	UNKNOWN	5

CHAPITRE 6

Périmètre, méthodologie et limites

Audit statique du code source, historique Git compris, en lecture seule. Aucun code n'a été exécuté et aucun environnement en fonctionnement n'a été sollicité.

OBJET	Portail client · muster-kundenportal (bundle Git)
COMMIT	`e0e3cde972c5` (main, 20/10/2026)
PÉRIODE	1er novembre 2026
CHARGE	1 jour-homme

DOMAINES AUDITÉS

DOMAINE	COUVERTURE	APPRÉCIATION
Authentification et sessions	● Complète	
Autorisation et cloisonnement des clients	● Complète	
Injection, entrées, appels externes	● Complète	
Secrets et cryptographie	● Complète	y compris tout l'historique Git
Journalisation et protection contre les abus	● Complète	
Configuration, conteneurs et CI/CD	● Complète	serveur et nginx hors du dépôt
Qualité du code, complexité, performance	● Complète	statique, sans mesure à l'exécution

CALCUL DE L'ÉVALUATION GLOBALE

Chaque domaine est évalué sur quatre niveaux : très bon (4), bon (3), actions requises (2), insuffisant (1). L'évaluation globale est la moyenne pondérée ; la sécurité compte pour 40 %, les autres domaines se partagent 60 %. Très bon à partir de 3,25 points, bon à partir de 2,5, actions requises à partir de 1,75, insuffisant en deçà. Si la sécurité est insuffisante, l'évaluation globale l'est aussi.

La sécurité découle des constats : insuffisant dès un constat critique, actions requises à partir de deux constats élevés ou de plus de dix constats moyens, très bon sans constat élevé et avec au plus trois constats moyens, bon dans les autres cas. Les autres domaines sont évalués par la revue sur la base des constats et de l'évaluation de la qualité.

ÉCHELLE DE GRAVITÉ

GRAVITÉ	CVSS	SIGNIFICATION
CRITIQUE	9,0–10,0	Exploitable avec un impact grave, agir immédiatement
ÉLEVÉ	7,0–8,9	Risque important, corriger à court terme
MOYEN	4,0–6,9	Risque sous conditions, planifier la correction
FAIBLE	0,1–3,9	Risque faible ou durcissement

LIMITES DE L'AUDIT

L'audit est un échantillonnage statique, limité dans le temps. Il ne remplace pas un test d'intrusion et ne prouve pas l'absence d'autres vulnérabilités.

L'évaluation porte sur le commit indiqué. La configuration et les données de l'environnement d'exécution n'entraient pas dans le périmètre.

HORS PÉRIMÈTRE

- Environnement en service, serveur, configuration nginx et contenu des bases de données
- Réglages hors du dépôt (protection de branche, secrets dans GitHub)
- Mesures à l'exécution – la performance est évaluée de façon statique

Je reste à votre disposition pour toute question.

Benjamin Barbé
Techspike SARL

Annexe

FICHIER	CONTENU
`Rohdaten_Beispiel.zip`	Résultats bruts de tous les outils (JSON, CSV) et vue d'ensemble
`Befunde_Beispiel.zip`	Tous les constats sous forme de fichiers JSON