

Bonobo Git Server

Sécurité, qualité, tests et migrabilité vers .NET 10



Sécurité



Fiabilité



Maintenabilité



Architecture



Tests



Migrabilité

Dépôt	github.com/jakubgarfield/Bonobo-Git-Server (licence MIT)
Version audité	commit 6277ecf, 5 mars 2021
Périmètre	17 130 lignes de C#, 198 fichiers, 331 tests
Plan d'action	27 actions numérotées, dont 13 sous 1 mois
Date	31 octobre 2026
Auditeur	Nicolas Cousin, Tech Lead .NET · nicolas-cousin.com

Cas d'étude réalisé sur un projet open source, à titre de démonstration de méthode. Il est réalisé sans lien avec les auteurs du projet. Les constats portent sur la version citée et ne préjugent pas des versions ultérieures.

1. Synthèse pour la direction

Bonobo Git Server est une application web ASP.NET MVC 5 (.NET Framework 4.6) qui héberge des dépôts Git sur un serveur Windows/IIS. Le code propre au projet est petit (17 000 lignes de C#), lisible, et doté d'une suite de 331 tests qui passent.

Trois risques demandent une action rapide :

1. **Mots de passe** : ils sont stockés avec un hachage rapide (SHA-512), et un ancien format MD5 est encore accepté. En cas de fuite de la base, ils se cassent en quelques heures (S-01, Critique).
2. **Composants vulnérables** : 6 bibliothèques ont des failles publiques, dont la couche d'authentification par cookie (Owin 3.0.1) et jQuery 1.12.4, servi à tous les utilisateurs (S-02, S-04).
3. **Plateforme hors support** : .NET Framework 4.6 n'est plus supporté depuis 2022, et le projet ne compile plus tel quel sur un poste à jour (M-01).

Ce qui va bien : pas de secret dans les 864 commits de l'historique, une couche d'accès aux données bien testée (57 %), et peu de défauts de fiabilité confirmés.

Le frein principal à toute évolution : les contrôleurs, le protocole Git et les filtres d'autorisation ne sont pas testés (0 %). Toute migration doit commencer par là.

Plan d'action : 27 actions numérotées P-01 à P-27 (section 4), dont 13 sous 1 mois.

Recommandation : 5 à 8 jours de corrections rapides (sécurité, passage en 4.8), puis une migration progressive vers .NET 10 (LTS), après avoir posé un filet de tests.

2. Notes par axe

Grille : **A** = aucun constat Majeur · **B** = constats Mineurs seulement, ou 1 Majeur isolé · **C** = plusieurs Majeurs · **D** = au moins 1 Critique, ou des Majeurs qui bloquent l'évolution · **E** = risque grave non maîtrisé, à traiter avant toute mise en production.

Axe	Note	Constats	En une phrase
Sécurité	D	S-01 (C), S-02 (M), S-04 (M), S-03 (m), S-05 (m)	Le hachage des mots de passe et les dépendances sont à corriger en priorité
Fiabilité	B	F-01 (m)	Peu de défauts confirmés par un moteur d'analyse récent
Maintenabilité	C	Q-01 (m), bruit de style (info)	Code lisible, mais bibliothèques copiées dans le dépôt et code mort
Architecture	C	A-01 (M)	Un contrôleur concentre trop de responsabilités
Tests	D	T-01 (M)	Suite saine mais aveugle sur l'autorisation et le protocole Git
Migrabilité	D	M-01 (M)	Plateforme hors support. La migration vers .NET 10 est un vrai chantier (System.Web, OWIN)

3. Feuille de route

Le détail action par action, numéroté, est en section 4.

Horizon	Actions	Effort estimé
Sous 1 mois	S-01 (PBKDF2/Argon2 + réinitialisation des comptes MD5) · S-02 (Owin 4.2, Newtonsoft.Json 13.0.x, DotNetZip remplacé par System.IO.Compression) · S-04 (jQuery 3.7) · <code>debug="false"</code> · passage en .NET Framework 4.8 (M-01) · S-03	5 à 8 j
1 à 3 mois	Tests des filtres d'autorisation et des contrôleurs (T-01) · SafeCombine + ordre validation, puis autorisation, puis disque (S-05) · découpage de <code>RepositoryController</code> (A-01) · nettoyage du dépôt (Q-01) · contrôle des vulnérabilités en CI · Newtonsoft.Json remplacé par System.Text.Json avec la montée d'IdentityModel	10 à 15 j
3 à 9 mois	Migration vers .NET 10 par étapes : bibliothèques en <code>netstandard2.0</code> , puis le site en ASP.NET Core (adaptateurs System.Web ou migration incrémentale), OWIN remplacé par le middleware d'authentification Core, EF6 sur .NET puis EF Core	30 à 50 j

4. Plan d'action : 27 actions

27 actions numérotées, dont **13** à mener sous 1 mois.

Une action est une recommandation distincte, localisée (fichier, composant ou paquet) et accompagnée d'un effort estimé. Garantie de la formule Flash : remboursement si le rapport contient moins de 10 actions. Ce rapport en contient 27.

N°	Fiche	Action	Où	Effort
Sous 1 mois · 13 actions				
P-01	S-01	Remplacer le hachage SHA-512 par PBKDF2-SHA256 (≥ 600 000 itérations) ou Argon2id, rehachage à la connexion	<code>Security/PasswordService.cs</code>	1j
P-02	S-01	Forcer la réinitialisation des comptes encore en MD5 après un délai annoncé	<code>PasswordService.cs</code> l. 59-64, base Users	0,5 j
P-03	S-02	Monter Microsoft.Owin et Owin.Security.Cookies en 4.2.x	<code>packages.config</code>	0,5 j
P-04	S-02	Monter Newtonsoft.Json de 9.0.1 en 13.0.x	<code>packages.config</code>	0,25 j
P-05	S-02	Remplacer DotNetZip (abandonné) par System.IO.Compression pour l'export zip	<code>RepositoryController.cs</code> l. 385-417	0,5 j
P-06	S-03	Afficher le même message que le compte existe ou non (mot de passe oublié)	<code>HomeController.cs</code> l. 125-128	0,25 j
P-07	S-04	Passer jQuery 1.12.4 en 3.7 et vérifier les écrans	<code>App_Start/BundleConfig.cs</code> , <code>Scripts/</code>	1j
P-08	F-01	Corriger string.Format avec string[] (conversion covariante)	<code>RepositoryController.cs</code> l. 488	0,1j
P-09	F-01	Matérialiser les IEnumerable énumérés plusieurs fois	<code>RepositoryController.cs</code> l. 52/56, 471/507	0,25 j

N°	Fiche	Action	Où	Effort
P-10	F-01	Corriger le nom de paramètre et la valeur inutilisée	WindowsAuthenticationHandler.cs l. 136-139	0,1 j
P-11	F-01	Remplacer le catch vide par une journalisation, valider la configuration AD au démarrage	ADBackendPre6.0.0Models.cs l. 77, DiagnosticReporter	0,5 j
P-12	M-01	Passer la cible de .NET Framework 4.6 à 4.8 (projet et tests)	*.csproj, web.config	1 j
P-13	M-01	Mettre compilation debug="false" dans la configuration livrée	web.config l. 56	0,1 j
1 à 3 mois · 12 actions				
P-14	S-02	Monter System.IdentityModel.Tokens.Jwt et Microsoft.IdentityModel.*	packages.config	0,5 j
P-15	S-02	Remplacer Newtonsoft.Json par System.Text.Json (avec tests de relecture des fichiers existants)	ADBackendStore.cs, ReceivePackRecovery.cs	1,5 j
P-16	S-05	Créer SafeCombine(racine, nom) et l'utiliser à la place des 23 Path.Combine sur les dépôts	Controllers/, Git/	1 j
P-17	S-05	Valider le nom, puis contrôler les droits, puis seulement accéder au disque	GitController.cs l. 37, 76, 94, 111	0,5 j
P-18	S-05	Corriger la regex du nettoyeur de noms de fichiers (antislash, . et ..)	OneFolderRecoveryFilePathBuilder.cs l. 15	0,25 j
P-19	Q-01	Gérer les bibliothèques front via libman ou npm, retirer les fichiers jamais servis	Scripts/, Content/	1 j
P-20	Q-01	Retirer .vs/ et les configurations locales du dépôt, compléter .gitignore	racine	0,1 j
P-21	Q-01	Ajouter un contrôle des paquets vulnérables en CI (dotnet list package --vulnerable)	appveyor.yml / CI	0,5 j
P-22	A-01	Extraire de RepositoryController des services navigation, archive et droits	Controllers/RepositoryController.cs	3 j
P-23	T-01	Tests des filtres d'autorisation (WebAuthorize*, GitAuthorizeAttribute)	Attributes/, Security/	2 j
P-24	T-01	Tests d'intégration HTTP des contrôleurs Repository, Account, Home	Controllers/	4 j
P-25	T-01	Tests de caractérisation du protocole Git (ReceivePackParser)	Git/GitService/	2 j
3 à 9 mois · 2 actions				
P-26	M-01	Migrer les bibliothèques en netstandard2.0 / .NET 10	Data/, Security/, Git/	10-15 j
P-27	M-01	Migrer le site vers ASP.NET Core (.NET 10), OWIN vers le middleware Core	Bonobo.Git.Server	20-35 j

5. Méthode et outils

- JetBrains Rider 2026.2.2 (inspections) et jb inspectcode 2022.3 (SARIF, 1 202 alertes, dont environ 85 % de style)
- Analyse de vulnérabilités des paquets NuGet (Rider) · dotCover (couverture) · gitleaks (secrets, historique complet) · scc (volume, complexité)
- Lecture ciblée du code sur les zones à risque : authentification, chemins de fichiers, contrôleurs
- Chaque constat cite un fichier et une ligne, et peut être vérifié. Les alertes d'outils non confirmées à la lecture ont été écartées (voir F-01).

6. Mesures de référence

Mesure	Valeur	Source
Code C# propre au projet	198 fichiers, 17 130 lignes, complexité 1 111	scc
Alertes jb inspectcode	1 202 (environ 85 % de style)	SARIF
Paquets NuGet vulnérables	6 (dont 1 sans correctif)	Rider
Secrets dans l'historique	0 vrai secret (6 faux positifs, 864 commits)	gitleaks
Tests	331 (327 réussis, 4 ignorés), couverture 24 %	Rider, dotCover
Cible	.NET Framework 4.6 (hors support depuis 04/2022)	csproj

7. Fiches constat

Les fiches détaillées suivent, classées par axe.

S-01 Mots de passe stockés avec un hachage rapide (SHA-512) et un ancien format MD5 encore accepté

- **Axe** : Sécurité · **Sévérité** : Critique · **Effort** : M
- **Preuve** : `Security/PasswordService.cs` l. 26-34 (`SHA512(SHA512(sel + mdp + sel))`), sans itérations) et l. 59-64 (un hash MD5 non salé est accepté, puis mis à jour à la connexion)
- **Preuve complémentaire** : la mise à niveau des anciens sels est désactivée (code commenté dans `PasswordService.cs` l. 41-55), et son test est marqué `[Ignore]` (`EFMembershipServiceTest.cs` l. 75, `TestThatValidatingAUserWithOldStyleSaltUpgradesTheirSalt`). Les comptes anciens gardent donc leur ancien format.
- **Impact métier** : en cas de fuite de la base, la plupart des mots de passe se retrouvent en clair en quelques heures. Ce sont souvent les mêmes que ceux du domaine Windows.
- **Recommandation** : PBKDF2-SHA256 (≥ 600 000 itérations), bcrypt ou Argon2id. Rehacher à la connexion suivante, et forcer la réinitialisation des comptes encore en MD5 après un délai.

Actions de cette fiche

P-01 Remplacer le hachage SHA-512 par PBKDF2-SHA256 ($\geq 600\,000$ itérations) ou Argon2id, rehachage à la connexion (1 j, sous 1 mois)

P-02 Forcer la réinitialisation des comptes encore en MD5 après un délai annoncé (0,5 j, sous 1 mois)

S-02 Dépendances avec vulnérabilités connues, dont une sans correctif

- **Axe** : Sécurité · **Sévérité** : **Majeure** · **Effort** : M
- **Preuve** : `packages.config`, analyse de vulnérabilités Rider (31/10/2026)

Paquet	Avis	CVSS	Exposition dans Bonobo	Correction
DotNetZip 1.10.1	CVE-2024-48510, CVE-2018-1002205 (traversée de répertoire à l'extraction)	9.1 / 5.5	Faible : la bibliothèque sert seulement à <i>créer</i> des zip (<code>RepositoryController.cs</code> l. 385) et à la compression zlib. Aucune extraction n'est faite.	Paquet abandonné : remplacer par <code>System.IO.Compression</code>
Microsoft.Owin 3.0.1	CVE-2020-1045, CVE-2022-29117	7.5	Couche d'authentification par cookie	4.2.2 ou plus
Microsoft.Owin.Security.Cookies 3.0.1	CVE-2022-29117	7.5	Cookie d'authentification	4.2.2 ou plus
Newtonsoft.Json 9.0.1	CVE-2024-21907 (dénégation de service)	7.5	Faible : 5 appels <code>JsonConvert</code> sur des fichiers JSON locaux (stockage AD <code>ADBackendStore.cs</code> l. 104 et 146, récupération des pushes <code>ReceivePackRecovery.cs</code> l. 36 et 68, migrations AD)	Court terme : 13.0.x (aucun changement de code). Moyen terme : System.Text.Json , en même temps que la montée d'Owin et d'IdentityModel, qui dépendent aussi de Newtonsoft
System.IdentityModel.Tokens.Jwt 4.0.4	CVE-2024-21319 (dénégation de service)	6.8	Flux WS-Federation (optionnel)	version à jour
Selenium.WebDriver 3.3.0	CVE-2023-5590	7.5	Tests seulement , aucun impact en production	à monter lors de la maintenance

- **Impact métier** : des failles publiques et outillées existent sur la couche d'authentification. DotNetZip n'est plus maintenu et ne recevra aucun correctif.
- **Recommandation** : monter Owin (effort faible, sans changement d'API), passer Newtonsoft.Json en 13.0.x, remplacer DotNetZip par System.IO.Compression, et ajouter un contrôle automatique des vulnérabilités en CI.
- **Migration vers System.Text.Json (moyen terme), points de vigilance** : (1) les fichiers JSON déjà sur disque doivent rester lisibles (`PropertyNameCaseInsensitive`, formats de dates) ; (2) `ParsedReceivePack` et ses sous-objets ont des propriétés `private set` et des `IEnumerable` que System.Text.Json laisse **vides sans erreur** sans `[JsonInclude]` ou `[JsonConstructor]` ; (3) sur .NET

Framework 4.8, le paquet apporte System.Memory, System Buffers et d'autres, avec des redirections de liaison dans `web.config` ; (4) ce code est à 0 % de couverture. Le faire derrière une petite interface de sérialisation, avec des tests qui relisent de vrais fichiers de l'ancienne version.

- **Note** : une CVE n'est pas forcément exploitable. Le score de DotNetZip (9.1) est ramené à un risque faible parce que le chemin vulnérable (l'extraction) n'est pas utilisé. Montrer ce tri est un argument pour le rapport.

Actions de cette fiche

P-03 Monter Microsoft.Owin et Owin.Security.Cookies en 4.2.x (0,5 j, sous 1 mois)

P-04 Monter Newtonsoft.Json de 9.0.1 en 13.0.x (0,25 j, sous 1 mois)

P-05 Remplacer DotNetZip (abandonné) par System.IO.Compression pour l'export zip (0,5 j, sous 1 mois)

P-14 Monter System.IdentityModel.Tokens.Jwt et Microsoft.IdentityModel.* (0,5 j, 1 à 3 mois)

P-15 Remplacer Newtonsoft.Json par System.Text.Json (avec tests de relecture des fichiers existants) (1,5 j, 1 à 3 mois)

S-03 Le formulaire « mot de passe oublié » révèle si un compte existe

- **Axe** : Sécurité · **Sévérité** : Mineure · **Effort** : S
- **Preuve** : `Controllers/HomeController.cs` l. 125-128 (message d'erreur distinct si l'utilisateur est inconnu)
- **Recommandation** : afficher le même message dans tous les cas (« si le compte existe, un e-mail a été envoyé »).
- Le jeton de réinitialisation, lui, est correct : sel aléatoire, expiration à 1 h (l. 132-133).

Actions de cette fiche

P-06 Afficher le même message que le compte existe ou non (mot de passe oublié) (0,25 j, sous 1 mois)

S-04 jQuery 1.12.4 servi à tous les utilisateurs (vulnérabilités XSS connues)

- **Axe** : Sécurité · **Sévérité** : Majeure · **Effort** : M
- **Preuve** : `App_Start/BundleConfig.cs` l. 13 (`~/Scripts/jquery-{version}.js` vers `Scripts/jquery-1.12.4.js`)
- **Avis** : CVE-2015-9251, CVE-2019-11358, CVE-2020-11022 et CVE-2020-11023, toutes corrigées en 3.5.0 ou plus
- **Impact métier** : ces failles, côté navigateur, permettent des scripts intersites (XSS) si du contenu contrôlé par un utilisateur passe par les fonctions concernées. Or Bonobo affiche des noms de dépôts, des messages de commit et des README.
- **Recommandation** : passer à jQuery 3.7, ou retirer jQuery (le code propre au projet en dépend peu). Retirer aussi les bibliothèques jamais servies : `jquery-ui`, `MicrosoftAjax.debug.js` ...

Actions de cette fiche

P-07 Passer jQuery 1.12.4 en 3.7 et vérifier les écrans (1 j, sous 1 mois)

S-05 Chemins construits avec `Path.Combine` à partir du nom de dépôt, sans contrôle de confinement

- **Axe** : Sécurité · **Sévérité** : Mineure (défense en profondeur) · **Effort** : S
- **Preuve** : 20 appels `Path.Combine(UserConfiguration.Current.Repositories, <nom>)`, dont `GitController.cs` l. 224 (`GetDirectoryInfo`, alimenté par la route `{repositoryName}.git`) et `RepositoryController.cs` l. 108-109, 162, 569 et 707.
- **Ce qui protège aujourd'hui** :
 - à la création (web et push), le nom passe par `NameValidityRegex` `([\w\.-])*([\w])$` en correspondance complète, plus `FileNameAttribute` `..`, `/`, `\` et `:` sont refusés (`RepositoryModels.cs` l. 47-55 et 96-102) ;
 - l'accès Git vérifie `HasPermission` sur le nom enregistré en base (`GitController.cs` l. 59, 81, 99).
- **Ce qui reste fragile** :
 - `RepositoryIsValid(repositoryName)` touche le disque (`DirectoryInfo` + `Repository.IsValid`) **avant** le contrôle de droits, avec la valeur brute de l'URL (l. 37, 76, 94). Un nom absolu fait renvoyer à `Path.Combine` le second argument tel quel, et un `..` fait sortir du dossier racine. Aujourd'hui, ce sont surtout les filtres d'URL d'IIS (`:`, `..`) qui l'empêchent. À confirmer par un test d'intrusion sur une instance.
 - Aucun code ne vérifie que le chemin final reste sous `Repositories`.
 - `TryCreateOnPush` teste `directory.Exists` **avant** `NameIsValid` (`GitController.cs` l. 111-120).
 - **Bug** dans le nettoyeur de noms de fichiers de récupération : `OneFolderRecoveryFilePathBuilder.cs` l. 15, `new Regex("([\\/:*?\"<>|])")`. En C#, `\\:` devient le motif `\:`, c'est-à-dire un deux-points échappé : **l'antislash n'est jamais retiré**, et `..` non plus. Ce n'est pas exploitable aujourd'hui (le nom vient d'un dépôt validé en base), mais le code ne fait pas ce qu'il annonce.
 - Export zip (`RepositoryController.cs` l. 411 et 417) : noms d'entrées construits sans normalisation. Le risque est faible, puisque Git refuse `..` dans les chemins.
- **Recommandation** : une seule fonction `SafeCombine(racine, nom)` (à utiliser à la place des 23 appels) qui valide le nom, fait `Path.GetFullPath`, vérifie que le résultat commence par la racine des dépôts, et contrôle les droits **avant** tout accès au disque. Corriger la regex `([\\\/\: *? "<> |])`, refuser `.` et `..`, et ajouter des tests de caractérisation (ces zones sont à 0 % de couverture).

Actions de cette fiche

P-16 Créer `SafeCombine(racine, nom)` et l'utiliser à la place des 23 `Path.Combine` sur les dépôts (1 j, 1 à 3 mois)

P-17 Valider le nom, puis contrôler les droits, puis seulement accéder au disque (0,5 j, 1 à 3 mois)

P-18 Corriger la regex du nettoyeur de noms de fichiers (antislash, `.` et `..`) (0,25 j, 1 à 3 mois)

F-01 Fiabilité : quelques défauts ponctuels, pas de problème systémique

- **Axe** : Fiabilité · **Sévérité** : Mineure · **Effort** : S
- **Preuve** :
 - jb inspectcode 2022.3 signalait 12 `PossibleNullReferenceException` et 10 `PossibleInvalidOperationException`. **Le moteur de Rider 2026.2 ne les confirme pas** sur les 15 fichiers les plus complexes. Ce sont donc des faux positifs de l'ancien moteur, écartés.

- Défauts confirmés par Rider 2026.2 :

- `RepositoryController.cs` l. 488 : `string.Format(linksurl, m)` avec un `string[]` converti en `object[]`, ce qui peut lever une exception à l'exécution ;
 - `RepositoryController.cs` l. 52/56 et 471/507 : énumérations multiples d'un `IEnumerable`, qui provoquent une double requête ou un double parcours Git ;
 - `WindowsAuthenticationHandler.cs` l. 136 : `ArgumentNullException("RedirectUri")` avec un nom de paramètre inexistant, et l. 139 une valeur calculée puis jamais utilisée ;
 - code mort : `ReceivePackParser.cs` l. 103, 142, 175 et 216, `RepositoryBrowser.cs` l. 321, `ADHelper.GetUserPrincipal` jamais appelée (l. 204) ;
 - exceptions avalées : `catch` vide dans `ADBackendPre6.0.0Models.cs` l. 77.
- **Analyse** : les cas null et `InvalidOperationException` signalés dans `ADBackend` et `ADHelper` (l. 127, 207-241, 302) restent plausibles **en cas de mauvais paramétrage** (annuaire AD incomplet, attribut absent, chemin non configuré). Ils donnent alors une erreur 500 sans message exploitable.
 - **Recommandation** : corriger ces points lors du passage en 4.8 (une demi-journée), et valider la configuration au démarrage avec un message clair (le `DiagnosticReporter` existant peut servir de base).
 - **Leçon de méthode** : la version de l'outil change le résultat. Le rapport cite la version utilisée (Rider 2026.2.2).

Actions de cette fiche

P-08 Corriger `string.Format` avec `string[]` (conversion covariante) (0,1 j, sous 1 mois)

P-09 Matérialiser les `IEnumerable` énumérés plusieurs fois (0,25 j, sous 1 mois)

P-10 Corriger le nom de paramètre et la valeur inutilisée (0,1 j, sous 1 mois)

P-11 Remplacer le `catch` vide par une journalisation, valider la configuration AD au démarrage (0,5 j, sous 1 mois)

Q-01 Hygiène du dépôt : bibliothèques tierces copiées dans le code, fichiers d'IDE commités

- **Axe** : Maintenabilité · **Sévérité** : Mineure · **Effort** : S
- **Preuve** : scc, 92 666 lignes au total, dont seulement **17 130 lignes de C#** propres au projet. Le reste est surtout du JavaScript tiers (26 000 lignes : jQuery, jQuery UI, MicrosoftAjax), des scripts Git embarqués (`App_Data/Git`) et du CSS de thème. Le dossier `.vs/` (`.vs/config/applicationhost.config`) et une config IIS de test ont été commités.
- **gitleaks** (864 commits, 6 alertes) : **aucun vrai secret**. On trouve une `sessionKey` IIS Express (`.vs/...` et `Bonobo.Git.Server.Test/iissetup/applicationhost.config` l. 127), des exemples dans `changelog.md` et un identifiant de site public dans Font Awesome.
- **Recommandation** : passer les bibliothèques front par un gestionnaire de paquets (libman ou npm), ajouter `.vs/` au `.gitignore`, et retirer les fichiers de configuration locale.

Actions de cette fiche

- P-19** Gérer les bibliothèques front via libman ou npm, retirer les fichiers jamais servis (1 j, 1 à 3 mois)
- P-20** Retirer .vs/ et les configurations locales du dépôt, compléter .gitignore (0,1 j, 1 à 3 mois)
- P-21** Ajouter un contrôle des paquets vulnérables en CI (dotnet list package --vulnerable) (0,5 j, 1 à 3 mois)

A-01 RepositoryController concentre la logique de l'application

- Axe** : Architecture · **Sévérité** : Majeure · **Effort** : M
- Preuve** : c'est le fichier C# le plus complexe (complexité 81, 674 lignes, scc) et le plus signalé (45 alertes jb). Il mêle navigation git, génération de zip, droits et présentation. RepositoryBrowser.cs vient ensuite (68 / 309).
- Recommandation** : extraire des services (navigation, archive, droits) testables sans HTTP. C'est aussi un préalable à la migration vers ASP.NET Core.

Actions de cette fiche

- P-22** Extraire de RepositoryController des services navigation, archive et droits (3 j, 1 à 3 mois)

T-01 Tests : 24 % du code couvert, et aucun test sur les contrôleurs, le protocole Git ni les filtres d'autorisation

- Axe** : Tests · **Sévérité** : Majeure · **Effort** : L
- Preuve** : dotCover (Rider, 31/10/2026, cible forcée en 4.8). Bonobo.Git.Server : 24 % (1 508 instructions couvertes sur 6 286).

Zone	Couverture
Data (accès aux données)	57 %
Security (dont PasswordService 98 %, EFMembershipService 74 %)	57 %
Controllers (dont RepositoryController, AccountController, HomeController)	0 %
Git (protocole push/pull, ReceivePackParser)	0 %
Attributes (WebAuthorizeAttribute, WebAuthorizeRepositoryAttribute) et GitAuthorizeAttribute (6 %)	0 %
RepositoryBrowser, Owin.Windows	0 %

- Croisement complexité x couverture** (scc + dotCover) : 7 des 10 fichiers C# les plus complexes ont 0 % de couverture (RepositoryController 81, RepositoryBrowser 68, AccountController 67, ReceivePackParser 48, ADHelper 46, WindowsAuthenticationHandler 30, HomeController 26). Seuls les dépôts EF (EFRepositoryRepository 94 %, EFTeamRepository 92 %, EFMembershipService 74 %) sont testés.
- Contexte** : le dernier commit du dépôt a supprimé les tests d'intégration, jugés « pas fiables sur AppVeyor ».
- Le chiffre global de dotCover (38 %) inclut le projet de tests lui-même. Il faut citer 24 % (l'application seule).

- **Impact métier** : les tests existants protègent la couche de données, mais **pas ce qui décide qui a le droit de lire ou de pousser du code**. Une régression sur les droits d'accès passerait sans être vue, ce qui bloque toute migration sereine.
- **Recommandation** : commencer par des tests sur les filtres d'autorisation et `GitAuthorizeAttribute` (effort S, gain maximal), puis des tests d'intégration HTTP sur les contrôleurs avant la migration.
- **Résultat de l'exécution** : 331 tests, **327 réussis, 0 en échec, 4 ignorés**. Les 4 ignorés sont 2 tests `[Ignore]` joués pour SQLite et SQL Server : la mise à niveau des sels (voir S-01) et la suppression d'un rôle qui a encore des membres (`EFRoleProviderTest.cs` l. 139). Le comportement de ce second cas diffère selon la base, et son auteur doute que la fonctionnalité doive exister. La suite est saine, mais elle teste surtout les couches basses.

Actions de cette fiche

P-23 Tests des filtres d'autorisation (`WebAuthorize*`, `GitAuthorizeAttribute`) (2 j, 1 à 3 mois)

P-24 Tests d'intégration HTTP des contrôleurs `Repository`, `Account`, `Home` (4 j, 1 à 3 mois)

P-25 Tests de caractérisation du protocole Git (`ReceivePackParser`) (2 j, 1 à 3 mois)

M-01 Le projet ne compile plus sur un poste de développement à jour

- **Axe** : Migrabilité · **Sévérité** : **Majeure** · **Effort** : S (passage en 4.8) / L (.NET 8/10)
- **Preuve** : `TargetFrameworkVersion v4.6` dans les deux csproj. Rider : « .NET Framework 4.6 is required ... but not installed ». Le pack de ciblage 4.6 n'est plus fourni par Visual Studio 2022, et cette version est hors support depuis avril 2022.
- **Recommandation** : passer tout de suite en 4.8 (supporté avec Windows), puis migrer par étapes vers .NET 8/10 (MVC 5 vers ASP.NET Core, OWIN vers le middleware Core, EF6).

Actions de cette fiche

P-12 Passer la cible de .NET Framework 4.6 à 4.8 (projet et tests) (1 j, sous 1 mois)

P-13 Mettre `compilation debug="false"` dans la configuration livrée (0,1 j, sous 1 mois)

P-26 Migrer les bibliothèques en `netstandard2.0` / .NET 10 (10-15 j, 3 à 9 mois)

P-27 Migrer le site vers ASP.NET Core (.NET 10), OWIN vers le middleware Core (20-35 j, 3 à 9 mois)