VN-Ch14-3 : Budgets de performance mesurés et tests

Complément du chapitre 14, section 14.2.2

14.6. Les budgets de performance mesurés

Le chapitre 12 a converti le principe de sobriété en cinq budgets opposables, présentés au tableau 61 du chapitre 12. Cette section donne ce que nous avons mesuré nous-mêmes, et seulement cela. Les mesures ont été prises à partir d'une construction de production faite sur le poste de développement, avec les scripts du code source : le script de budget du premier chargement, le script de budget par adresse, qui ouvre la construction dans Chrome devant une API simulée et relève tout le JavaScript et le CSS chargés jusqu'à l'écran stable, le build du contenu pour les paquets, et le script Lighthouse du code source. Les tailles sont compressées en gzip, avec 1 Ko valant 1 000 octets, ce qui est la lecture la plus stricte. Aucune mesure n'a pu être faite sur un appareil réel, faute d'appareil physique disponible.

Le tableau VN-43 confronte chaque budget à ce qui a été mesuré. Une cible n'y figure jamais comme une mesure : lorsque rien n'a été mesuré, la case porte « non établi ».

Tableau VN-43 : Budgets du chapitre 12 et valeurs mesurées.

Budget (tableau 61 du chapitre 12)	Limite	Valeur mesurée par nous	Résultat
Premier chargement de l'application, compressé	moins de 250 Ko	186,54 Ko (script de budget du premier chargement)	Dans le budget
Premier chargement et fragments, par adresse	moins de 250 Ko par adresse	41 adresses mesurées ; maximum 288,54 Ko sur /enseignant/cours ; 259,84 Ko sur /design-system ; 248,80 Ko sur la fiche d'un chapitre élève ; 246,62 Ko sur l'accueil élève ; toutes les autres adresses sous 247 Ko	Dans le budget, sauf deux exceptions écrites à 300 Ko
Paquet d'un chapitre, médias compris	moins de 40 Mo	0,75 Mo, 1,69 Mo et 0,54 Mo pour les trois chapitres ; capsules publiées hors paquet : 14 pour 95,7 Mo	Dans le budget pour les paquets
Une ressource de simulation	moins de 3 Mo	Plus gros module de l'inventaire : 249 532 octets ; plus gros des 30 modules du code source : 275 420 octets, avant compression	Dans le budget
Première interaction, Android d'entrée de gamme	moins de 4 s	Appareil réel : non établi. Émulation Lighthouse mobile (processeur ralenti 6 fois, latence 300 ms, 700 kbit/s) sur six écrans : LCP de 6,79 s à 14,43 s, FCP de 5,01 s à 5,75 s, blocage total de 0 à 82 ms, score de performance de 61 à 64	Non établi (appareil) ; émulation fournie, au-delà de 4 s pour le LCP
Mémoire pendant une simulation	moins de 300 Mo	Non établi	Non établi

Note. Le LCP émulé est celui que Lighthouse calcule sous ralentissement simulé d'une connexion et d'un processeur ; il n'est pas la première interaction du budget, et une émulation n'est pas un appareil. Mesures faites sur l'API simulée du code source et non sur le serveur déployé.

La figure VN-31 donne, adresse par adresse, le poids chargé jusqu'à l'écran stable. Elle regroupe les adresses d'un même espace dont le poids est identique, par exemple les cinq écrans de la console du chercheur, d'où 29 lignes pour 41 adresses.

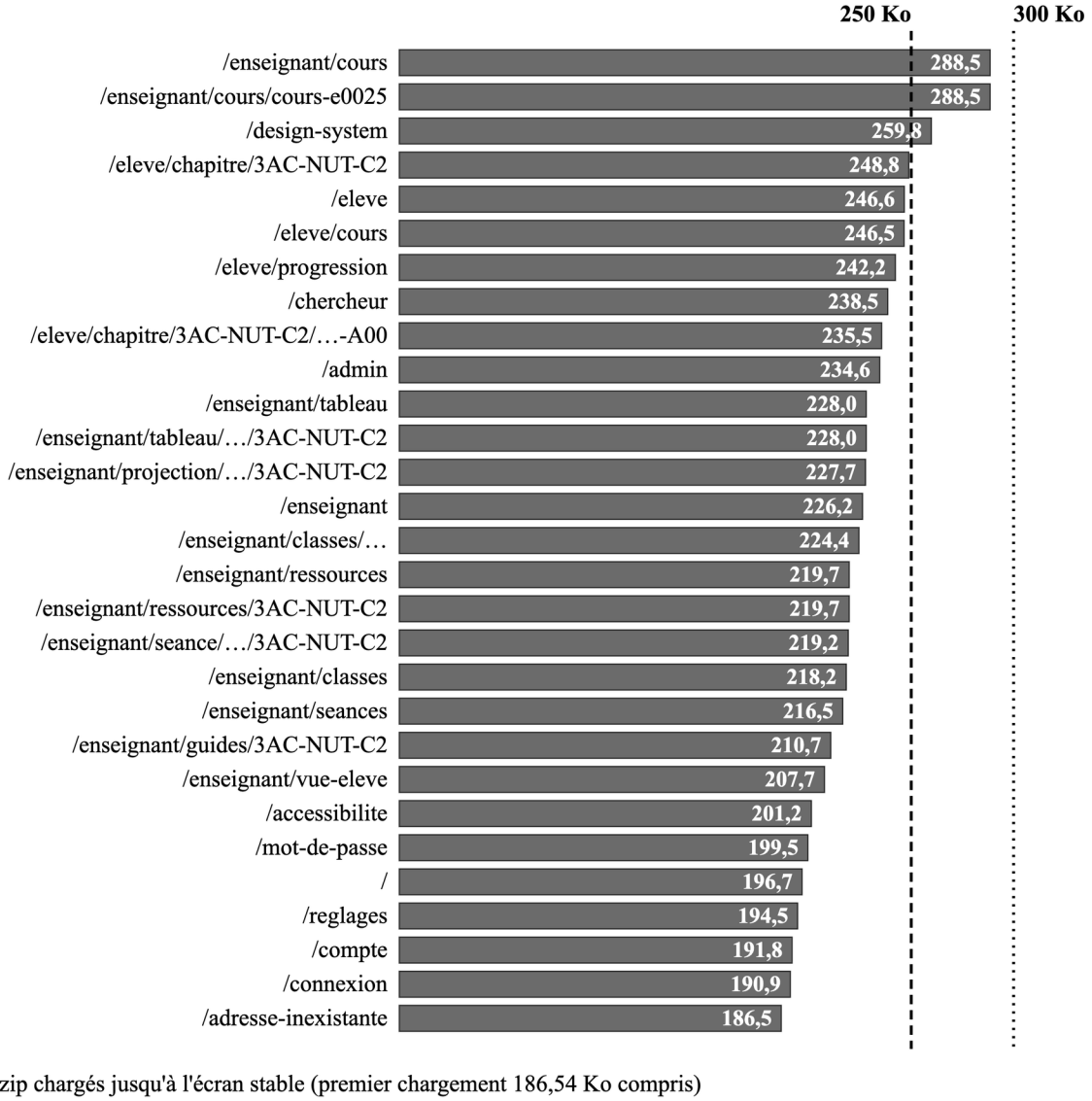


Figure VN-31 : Poids gzip, en kilooctets, du JavaScript et du CSS chargés jusqu'à l'écran stable pour 29 groupes d'adresses du dispositif ; les deux traits verticaux marquent le budget de 250 Ko et l'exception écrite de 300 Ko.

Dans la figure VN-31, les adresses de l'enseignant comptent parmi les plus lourdes, l'éditeur de cours chargeant à lui seul 102,00 Ko de fragments à la demande, et il est le seul écran de ce rôle à dépasser le budget général, sous une exception écrite que la conception justifie par le fait qu'il s'agit d'un écran de bureau hors du périmètre des téléphones des élèves. La page interne du

système de design dépasse aussi 250 Ko, sous la même limite exceptionnelle. Enfin, les écrans de l'élève restent entre 187 et 249 Ko, avec une marge étroite sur la fiche de chapitre, à 1,2 Ko de la limite, et la route de lecture d'une activité à 235,54 Ko dépasse la cible interne de 230 Ko que la conception s'était donnée pour elle, sans dépasser le budget général.

Le script Lighthouse du code source, dans ses conditions habituelles (préréglage mobile, processeur ralenti 4 fois, connexion de 4G lente), a donné un LCP de bureau de 0,73 à 1,46 s, un décalage cumulé de mise en page de 0,000 à 0,001 et un score d'accessibilité de 100 pour six écrans, tous installables ; le score de performance mobile émulé va de 72 à 89 et le LCP mobile émulé de 3,27 s à 6,63 s. Nous l'avons relancé en durcissant les conditions, selon la demande d'une émulation d'appareil d'entrée de gamme : processeur ralenti 6 fois, latence de 300 ms et débit de 700 kbit/s. Le score de performance tombe à 61 à 64, le LCP à 6,79 s pour la page de connexion, 8,73 s à 8,75 s pour les accueils, et 14,43 s pour le lecteur et le tableau de bord de l'enseignant ; le FCP va de 5,01 s à 5,75 s et le blocage total de 0 à 82 ms. Le score d'accessibilité reste à 100 et le décalage de mise en page à 0,001 au plus. Cette émulation dit que, sur un processeur six fois plus lent et un réseau de ce débit, les écrans lourds dépassent largement quatre secondes avant d'afficher leur plus grand élément ; elle ne mesure pas la première interaction, et elle ne remplace pas la mesure sur un Android réel, qui reste non établie. Le script signale aussi un audit d'accessibilité en échec sur le lecteur, la règle sur la cohérence entre l'étiquette visible et le nom accessible, malgré le score de 100.

Les tests du monorepo ont été exécutés. Les tests unitaires comptent 3 781 tests dans six paquets (184 pour les schémas, 1 493 pour les simulations, 635 pour le système de design, 124 pour le contenu, 249 pour l'API et 1 096 pour l'interface). Les tests d'intégration de l'API, joués sur une base PostgreSQL 16 migrée, comptent 164 tests dans 18 fichiers et ont tous réussi. Les parcours de bout en bout, joués sur une construction de production avec Chrome, comptent 126 tests dans 17 fichiers : 120 ont réussi, 6 ont été ignorés (captures conditionnées à une variable d'environnement) et aucun n'a échoué. Ces résultats disent que la suite du code source passe sur ce poste ; ils ne disent pas que les tests couvrent ce qu'il faudrait couvrir. La section suivante revient aux écrans, pour ceux que la version 4 a ajoutés.