Grenoble INP - Ensimag
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Rapport de stage assistant ingénieur

Effectué à E-Aubonne

Conception, développement et intégration d'un module énergétique–financier photovoltaïque unifié dans l'outil d'étude de panneaux solaires d'E-Aubonne

TONDE NDEBIA Junior Gael

2e année – Filière Ingénierie des Systèmes d'Information

10 Juillet – 04 Aout 2025(8 semaines)

E-Aubonne
40 Av. Alsace-Lorraine
95600 Eaubonne

Responsable en entreprise : CAMATCHY Anthony

Tuteur de l'école : FRENOY Antoine

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

- Contexte général

La montée en puissance du photovoltaïque et la pression pour accélérer la transition énergétique rendent cruciale la phase de pré-étude : quantifier rapidement un gisement de production, estimer la production d'énergie des photovoltaiques et disposer d'indicateurs financiers fiables. Or ces évaluations restent souvent dispersées entre outils hétérogènes et manipulations manuelles.

- Problématique

Cette fragmentation allonge les délais, multiplie les risques d'hypothèses incohérentes (valeurs limites non validées, conversions oubliées, …) et complique la traçabilité (difficile d'expliquer comment une estimation ou un flux de trésorerie a été produit). L'absence d'un socle unifié ralentit la décision technique et financière, ce problème ne fera que s'empirer avec d'éventuelles expansion de l'entreprise. N'y a-t-il pas une solution permettant de centraliser tous les besoins d'estimation (de production de production et de rentabilité financière) des systèmes photovoltaique ?

- Objectifs spécifiques

Ainsi, notre projet de stage a pour objectifs de concevoir, intégrer et analyser les données d'estimation de production de tous les types de système photovoltaïque (couplé au réseau, hors réseau et suiveur) dans l'application interne de l'entreprise ; de réduire la complexité de préparation d'un rapport d'estimation ainsi que les risques d'erreurs manuelles dus à l'usage d'outils multiples pour l'estimation (PVGIS et AutoCalSol si le système pv est de type suiveur ou de type hors réseau) ; de modéliser les données et calculs puis implémenter les indicateurs financiers (cash-flow, etc.) ; de créer une interface ergonomique, valider les entrées et orchestrer via des contrôleurs fiables ; enfin, d'industrialiser la qualité: tests unitaires, couverture ciblée, analyse statique, documentation technique fournie.

PARTIE 1 : CADRE DE REFERENCE

CHAPITRE I : DESCRIPTION DE L'ENVIRONNEMENT AUTOUR DU PROJET

Dans cette partie, il est question de mettre en avant l'environnement qui encadre le thème ainsi que les notions y afférent. De ce fait, nous allons présenter l'entreprise E-Aubonne, puis les motivations qui ont conduit au choix du développement du thème, et enfin le vocabulaire nécessaire à la compréhension de la problématique qu'il traite.

I.1. Présentation de E-Aubonne

E-Aubonne est une startup récente basée à Eaubonne (Val-d'Oise) [1]. Elle opère dans le secteur de l'installation électrique spécialisée, avec une forte orientation vers les infrastructures de recharge et la transition énergétique. E-Aubonne a une organisation juridique particulière, propice au développement des startup.

I.1.1. Organisation générale

E-Aubonne est une SASU (Société par Actions Simplifiée Unipersonnelle). Une SASU est une forme juridique d'entreprise en France dans laquelle la société est constituée d'une seule personne (physique ou morale) qui est actionnaire unique et dirige l'entreprise ou désigne un président. De ce fait, E-Aubonne n'est constituée que d'un seul employé, son Président Directeur Général qu'est M. Anthony Camatchy.

I.1.2. Services

Cette startup fournie des services variés dans diverses branches des domaines électrique et électroniques pour particuliers, copropriétés et entreprises. Nous pouvons citer entre autres :

- L'installation et la maintenance de bornes de recharge pour voitures électriques à domicile, au travail, pour des collectivités et pour des voiries publiques ;
- L'installation et la maintenance des systèmes photovoltaïques ;
- L'installation et la maintenance électriques (prises, interrupteurs, systèmes électriques complet du bâtiment, …) ;
- La configuration des équipements électroniques tels que les systèmes d'alarmes, les volets roulants et les pompes à chaleur.

I.1.3. Certifications et accréditations de l'entreprise

L'entreprise dirigée par M. Camatchy dispose de nombreuses habilitations et certifications lui donnant le droit d'exercer les services su-citées sur l'ensemble du territoire. On a entre autres :

- Des habilitations électriques : le président de E-Aubonne est Titulaire d'habilitations électriques reconnues au niveau national, garantissant sécurité, conformité aux normes et qualité dans toutes les installations.
- Des certifications IRVE (Infrastructure de Recharge de Véhicules Électriques) comme :
    - IRVE P1 : Installation de bornes basiques ≤ 22 kW (maisons individuelles).
    - IRVE P2 : Installation de bornes communicantes ≤ 22 kW (commerces, copropriétés, collectivités).
    - IRVE P3 : Installation de bornes rapides > 22 kW en courant continu (CHAdeMO / Combo2).

I.2. Définitions générales

I.2.1. Système Photovoltaïque [2] :

Le photovoltaïque désigne la conversion directe de l'énergie lumineuse du soleil en électricité grâce à des cellules semi-conductrices (souvent en silicium). Un système photovoltaïque (en abrégé « pv » ou « système pv ») complet comprend les modules (panneaux), les structures de support, le câblage, éventuellement des dispositifs de stockage (batteries) et une électronique de puissance (onduleur, régulateur). Il en existe 3 types :

- Système couplé réseau (Grid) : Les panneaux injectent l'énergie produite dans le réseau public via un onduleur synchronisé. Pas de stockage local (ou facultatif). L'utilisateur consomme une part en direct (autoconsommation) et le surplus est injecté (vente ou compensation).
- Système hors réseau (Off-grid) : Fonctionne sans connexion au réseau. Nécessite un stockage (batteries) et un dimensionnement spécifique (profil de charge, autonomie, profondeur de décharge). Il assure la continuité d'alimentation d'un site isolé.
- Système suiveur (Tracker) : Identique à un système PV classique (grid ou off-grid) mais les modules pivotent (1 ou 2 axes) pour optimiser l'angle d'incidence solaire et accroître la production annuelle (gain typique +10 à +30% selon latitude).

I.2.2. PVGIS (Photovoltaic Geographical Information System) :

Développé par le Centre Commun de Recherche de la Commission Européenne [3], PVGIS est un outil en ligne permettant : D'estimer la production d'énergie photovoltaïque des trois types de systèmes, en tenant compte de la localisation géographique, de la météo et des caractéristiques techniques des panneaux et de fournir des résultats détaillés (production mensuelle, annuelle, histogrammes de l'état de charge, etc.) sous forme de rapports, graphiques, CSV ou JSON. PVGIS dispose aussi d'une API [4] pour automatiser les calculs dans des applications externes. Voici quelques caractéristiques de cette API :

- Accès REST (requêtes HTTP GET paramétrées)
- 2 Endpoint principaux : PVcalc (estimation production des pv couplés au réseau et des pv suiveurs) et OffGrid (estimation production des pv couplés au réseau). Exemple d'URL d'appel à l'API : https://re.jrc.ec.europa.eu/api/v5_2/PVcalc?lat=46.5&lon=6.6&peakpower=5&loss=14
- Réponse JSON structurée contenant les séries temporelles agrégées (mensuelles), les totaux annuels, les paramètres d'entrée normalisés et parfois des histogrammes pour l'irradiation ou la production.

I.2.3. AutoCalSol

C'est un outil en ligne développé par **l'INES (Institut National de l'Énergie Solaire, France) [5]** et destiné à : Calculer la **production d'énergie solaire photovoltaïque** pour une installation donnée en France et en Europe ; intégrer des données et hypothèses financières pour estimer la

viabilité d'un système photovoltaïque sur le long terme ; Générer des rapports utiles aux bureaux d'études, installateurs et chercheurs.

CHAPITRE II : ANALYSE DE LA PROBLEMATIQUE ET ELABORATION DE LA SOLUTION

II.1. PROBLEME A RESOUDRE

L'entreprise dispose déjà de divers outils hétérogènes pour l'étude et l'installation de panneaux photovoltaïques. En particulier pour l'estimation de production et l'estimation financière, l'on utilise PVGIS, AutoCalSol, Microsoft Word, Microsoft Excel, etc. La pluralité et la non-centralisation de ces services transforment les études photovoltaïques pour un seul client en une tache fastidieuse et chronophage. Le problème à résoudre ici est de trouver un moyen de centraliser les services d'estimation de production et d'estimation financière dans une même application interne tout en réduisant les multiples étapes manuelles d'étude photovoltaïque pour chaque client.

II.2. ETAT DE L'ART

Dans cette partie, nous allons étudier les processus et solutions utilisées par E-Aubonne jusqu'à présent pour résoudre le problème posé ci-dessus (mutualiser les taches d'estimation de production et d'estimation financière pour les systèmes pv et réduire les étapes).

II. 2.1. Description de l'existant

Jusqu'à peu, les estimations de production d'énergie des systèmes photovoltaïques étaient réalisées via le site PVGIS. Pour les projets qui nécessitaient aussi une estimation financière on utilisait AutoCalSol. Par suite, nous présenterons le rôle des outils PVGIS et AutoCalSol dans le processus de prévision de la production et de la rentabilité des systèmes photovoltaïques que l'on souhaite concevoir pour un client donné.

II. 2.1.1. Outils utilisés

- PVGIS : Le président de E-Aubonne utilise principalement cet outil pour l'estimation de la production énergétique des panneaux photovoltaïques dans plusieurs de ses projets. On l'utilise en particulier dans des clients demandant une comparaison de production entre un système couplé et un système hors réseau. Ou simplement s'il faut faire des estimations de production pour les systèmes pv hors-réseau ou pv suiveur car l'outils suivant ne prends pas en compte ces types de pv.
- AutoCalSol : Pour les clients qui demandent des informations plus détaillées sur la rentabilité de l'installation de panneaux solaire, l'on utilise AutoCalSol (et non pas PVGIS) pour l'estimation de production et l'approximation de la rentabilité dans le temps du système photovoltaïque. On l'utilise en priorité pour les systèmes pv couplés au réseau.

II. 2.1.2. Processus actuel

Le processus actuel d'estimation de production énergétique des système photovoltaïques pour un client donné d'E-Aubonne se fait comme suis :

- Si le système photovoltaïque à estimer est de type couplé au réseau, on utilise l'outils AutoCalSol : car il permettra directement de faire l'estimation financière après l'estimation de production.
  i. On remplit les données d'estimation de production, puis les données financières (prix du Kilowatt, Investissement de départ, …) du système de notre client dans les formulaires AutoCalSol
  ii. On exporte ensuite en PDF et csv les résultats d'estimation de production et financière fournis par AutoCalSol
  iii. Enfin, on rassemble manuellement les résultats dans Word et Excel pour fournir un rapport au client. Ce rapport contient les estimations de production d'énergie du système photovoltaïque du client sur quelques années en plus de graphes financiers qui montrent à partir de combien d'années le système sera rentable financièrement.
- Sinon on utilise l'outils PVGIS pour l'estimation de production (car AutoCalSol n'intègre pas les estimations pour les systèmes pv hors-réseau et pv suiveur)
  i. On remplit les données du système de notre client dans le formulaire PVGIS
  ii. On exporte ensuite en pdf et csv les résultats d'estimation de production fournis par PVGIS
  iii. Ensuite on se rend dans AutoCalSol pour remplir les formulaires et avoir des données sur la rentabilité du projet. On fait quelques approximations en entrées des formulaires afin d'avoir une estimation financière assez correcte meme avec des systèmes photovoltaïques hors réseau ou suiveur (en temps normal, AutoCalSol ne fait des estimations que sur les systèmes PV coulpes au réseau électrique)
  iv. Par suite, on exporte en pdf et csv les résultats d'estimation de production et financière fournis par AutoCalSol
  v. Enfin, on rassemble les résultats manuels dans Word et Excel pour fournir un rapport au client. Ce rapport contient les estimations de production d'énergie du système photovoltaïque du client sur quelques années (issus de PVGIS) en plus de graphes financiers (issus d'AutoCalSol) qui montrent à partir de combien d'années le système sera rentable financièrement.

II. 2.2. Analyse de l'existant

Dans cette partie, nous allons étudier les systèmes et processus déjà en place au moyen d'une matrice SWOT. Commençons par définir ce qu'est une matrice SWOT

Une **matrice SWOT** est un outil d'analyse stratégique qui permet d'évaluer une organisation, un projet ou une idée en identifiant : **S** (Strengths) qui représente les **forces** internes ; **W** (Weaknesses) pour les **faiblesses** internes ; **O** (Opportunities) pour les **opportunités** externes et enfin **T** (Threats) pour parler des menaces externes.

Voici la matrice SWOT qui ressort de notre systèmes et processus existants décrits plus haut :

Strengths (Forces)	Weaknesses (Faiblesses)
❖ Les Outils reconnus (PVGIS, AutoCalSol) et donc les résultats sont crédibles. ❖ Les outils sont disponibles facilement (web) et sont simples à utiliser.	❖ Hétérogénéité des hypothèses entre les outils AutoCalSol et PVGIS pour les estimations de production et de rentabilité des

❖ Avec le fonctionnement actuel, on a une Flexibilité de mise en forme dans Word/Excel.	panneaux solaires de type hors réseau et suiveur : absence de référentiel central. ❖ Traçabilité limitée (versions, sources, unités), reproductibilité faible. ❖ Temps de production élevé car il faut naviguer entre plusieurs outils pour les estimations d'un seul client : non industrialisable.
Opportunities	Threats
❖ Automatisation via API gratuite de PVGIS et moteur financier unifié. ❖ Existence d'une application JAVA interne pour les systèmes photovoltaïques (qui fait des pré-études sur les matériaux nécessaires pour la construction de panneaux solaires mais rien n'est fait pour l'estimation de production) ❖ Standardisation des hypothèses d'estimation de production d'énergie des photovoltaïques et des gabarits de rapports en utilisant un seul outil externe (l'API PVIS).	❖ Évolutions d'interface des outils tiers, notamment AutoCalSol qui peut voir son accès restreint ou les couts de licence augmenter à tout moment (c'est un outil privé et non pas une initiative de l'UE comme PVGIS). ❖ Perte de savoir-faire si processus reste tacite et manuel.

Tableau 1 : matrice SWOT

II. 2.3. Critiques de l'existant

L'analyse de la matrice SWOT précédente nous a permis de ressortir les limites des éléments du processus d'estimation d'énergie/financière d'un client ainsi que les impacts de ces limites.

- ❖ Limites majeures
  - o Processus à forte charge manuelle.
  - o Données diverses entre outils, ce qui rend la consolidation fragile.
  - o Difficile à faire évoluer par rapport au contexte spécifique des client d'E-Aubonne (nouveaux scénarios, métriques, gabarits).
- ❖ Impacts

  Les impacts principaux sont les coûts de temps élevés et les risques de non-qualité (erreurs, incohérences du fait des approximations pour du processus les photovoltaïques qui ne sont pas couplées au réseau).

Somme toute, l'existant fournit des résultats crédibles mais au prix d'un flux manuel dispersé, peu traçable et coûteux. Une solution intégrée (collecte PVGIS, calculs énergétiques et financiers unifiés, génération automatique de rapports) est nécessaire pour fiabiliser, accélérer et industrialiser le processus.

II.3. PROPOSITION DE SOLUTION

II.3.1. Formulation du thème

Dans le souci de pallier aux limites des processus d'estimation soulevés dans la partie précédente et des opportunités ressorties dans la matrice SWOT, nous avons opté pour une solution informatique, robuste et intégrée dans l'application Java d'étude photovoltaïque dont E-Aubonne dispose déjà. Notre stage assistant ingénieur se fera donc selon le thème suivant :

« Conception, développement et intégration d'un module énergétique–financier photovoltaïque unifié dans l'outil d'étude de panneaux solaires d'E-Aubonne ».

II.3.2. Cahier de charges

II.3.2.1. Expressions des besoins

- Besoins fonctionnels

Ce sont les fonctionnalités (cas d'utilisation) à développer. Ce système doit permettre de :

- Estimer la production d'énergie de tout type de PV (couplé au réseau, hors réseau, suiveur) ;
- Voir les graphes d'estimation de production d'énergie des PV ;
- Voir les graphes de rentabilité financière du système PV ;
- Exporter les résultats tant financiers que de production d'énergie en CSV ;
- Exporter les résultats tant financiers que de production d'énergie en PDF ;

- Besoins non fonctionnels

Il s'agit de :

- Sécurité et communication : Comme le module d'estimation échangera avec une API externe (PVGIS) il faut s'assurer que c'est échanges externes soient chiffrés et que toutes ls erreurs réseau soient bien gérées ;
- Fiabilité, robustesse et résilience : toutes les entrées de formulaires doivent être systématiquement validées et on doit avoir une tolérance aux réponses partielles ou erronées de l'API,
- Performance et réactivité : Il s'agit d'avoir une UI réactive dans laquelle d'interface ne plante pas lorsqu'on effectue des taches ou des calculs lourds. Aussi il faut avoir une certaine réactivité réseau pour l'appel API ;
- Évolutivité et Maintenabilité : Le système doit être conçu avec une architecture modulaire, testable, et maintenable, permettant l'ajout aisé de nouveaux composants ;

II.3.2.2. Déroulement du projet

Afin de mener à bien notre projet, il est important pour nous de planifier les différentes tâches qui nous incombent notamment :

- L'interview : cette étape est primordiale car elle permet de définir clairement les besoins et les attentes du produit. Dans notre cas, cela s'est fait en une réunion d'une heure durant laquelle mon maitre de stage me présentait l'application java interne de l'entreprise ainsi que ses soucis actuels concernant l'estimation de production ;
- L'élaboration du cahier de charge et architecture de la solution : il s'agit pour nous de produire un document qui permet d'expliquer toutes les spécificités, les attentes et les contraintes de notre projet. Après avoir analysé les besoins de l'entreprise et les processus existants nous avons proposé un cahier de charge qui réponds aux préoccupations de l'entreprise. Ce cahier de charge fut validé. Pour l'architecture, nous avons produit un diagramme UML de composants (*voir annexe 4*). De plus, notons que cette partie et l'interview se sont faites les deux premières semaines de stage ;
- Implémentation du projet (3e-6e semaines de stage) : Cette partie correspond à l'écriture à proprement dit du code.
- Déploiement, tests et debug de l'application (7e et 8e semaines de stage) : durant cette phase, les tests furent écrits, l'application fut déployée en fichier .jar et elle fut améliorée en fonctions des feedbacks de l'utilisateur (président d'E-Aubonne).

PARTIE 2 : METHODOLOGIE ET MISE EN ŒUVRE DU TRAVAIL

CHAPITRE III : ARCHITECTURE GLOBALE DU SYSTEME

III.1. CHOIX D'ARCHITECTURE

Pour les modules que nous devons ajouter à l'application de l'entreprise, nous avons choisi une architecture MVC (Model Vue Controller) enrichie d'une couche de Service. C'est une architecture logicielle constituée de 4 couches principales :

- Modèle (Model) : porte les données et règles métiers locaux
- Vue (View) : interface utilisateur (dans notre cas en Swing) qui affiche formulaires, tableaux et graphiques, sans logique métier.
- Contrôleur (Controller) : orchestre les actions de l'utilisateur.
- Services : Elle contient la logique métier directement utilisable par le contrôleur. Dans notre cas, c'est une couche applicative dédiée aux appels externes (PVGIS), au parking, aux calculs financiers et à l'export ; elle encapsule la complexité (HTTP, tolérance JSON, timeouts/retries, PDF/CSV) et expose des API simples aux contrôleurs.

Nous avons fait ce choix pour plusieurs raisons :

- Le code java de l'application dans laquelle on doit intégrer les fonctionnalités d'estimation avait déjà des modèles et un contrôleur. Ainsi contrôleur il est plus simple de réorganiser le code en une architecture MVC avec des Services (pour rendre les contrôleurs moins « lourds » et plus testables).
- On a une séparation claire des responsabilités ce qui rend le code plus lisible et maintenable.
- La testabilité élevée des calculs, parsing et appels API sans forcément passer par l'UI.
- La robustesse de l'UI : les opérations longues sont dans des services et les contrôleurs (UI non bloquée).
- Nous avons une meilleure extensibilité. Par exemple, on peut ajouter de nouvelles stratégies d'export ou sources de données sans changer l'UI.
- Enfin, cette architecture améliore la réutilisabilité car la même logique service pourra être intégrée dans une future application interne (sans la Vue).

Dans la suite, nous montrerons détaillerons le contenu de chacun des 4 principaux packages par rapport à notre étude en particulier.

III.2. DETAILS DES PACKAGES

Maintenant, nous allons détailler le rôle de chaque package de notre application.

- Le package Modele pour encapsuler les données des différents formulaires et réponses de l'API PVGIS. Il a deux principaux sous-packages
  - Le package Modele.pvgis : il permet de définir les DTO (Data Transfer Object ou objet de transfert de données sont des classes dont le rôle est de transporter des données entre couches) d'entrée/sortie pour les estimations PV (Grid/Tracker/Off-Grid). Aussi, il sert d'interface de données pour les services liés à pvgis et pour l'UI (les formulaires).
  - Le package Modele.finance : Son rôle est d'encapsuler les hypothèses économiques et les résultats calculés pour la prise de décision.
- Le package Vue dans lequel on a créé deux packages pour notre étude
  - Le package Vue.estimationProd : qui contient les codes principaux des 3 interfaces d'estimation de production des différents types de PV (couplés au réseau, hors réseau et suiveur).
  - Le package Vue.util et Vue.ui : ils ont été créés pour réduire les redondances et factoriser le code. Ils contiennent respectivement les utilitaires et les styles communs à toutes les pages.
- Le package Controleur

- Le package controleur.estimationProd : Il encapsule les actions en arrières plan (asynchrones et synchrones) des différentes actions d'estimation et d'export. Pour se faire il utilise directement les méthodes du package Service.

❖ Le package Service qui implémente la logique métier (estimation de production d'énergie et financière) de l'application avec les exports. Nous avons essentiellement travaillés dans ces 3 sous-packages :

- Le package service.pvgis : pour l'accès résilient à l'API PVGIS (HTTP), construction d'URL, parsing JSON vers les modèles.
- Le package service.finance : pour les calculs économiques à partir des résultats PV. Il faut noter qu'après quelques recherches, nous avons proposés à l'entreprises certains indicateurs pertinents pour connaitre la rentabilité des systèmes photovoltaïques au fils des ans. Ces indicateurs sont : Le Cash-flow cumulé par an ; les recettes vs les dépenses par an ; le VAN par an et le VAN total ;
- Le package service.export : qui implémente des exports des résultats en fichiers (PDF/CSV) via les patrons Stratégie et Façade.
- Le package service.validation : dont le rôle est la validation centralisée des entrées (bornes/formats) afin de ne pas surcharger les codes des interfaces et de faciliter les tests.

❖ Le package Test : il regroupe tous les tests de l'application

❖ Le package Main : il contient le point de départ de l'application (fichier Main.java)

Ainsi, nous avons construit ce projet sous une architecture MVC avec Service. Nous avons choisi cette architecture sur la base de critères divers afin qu'elle puisse parfaitement seoir aux modules que nous développons. Notons qu'un diagramme de composants qui traduit les interactions entre tous les packages est en annexe (*voir annexe 4*). Par suite, nous donnerons les détails d'implémentation.

CHAPITRE IV : ETUDE TECHNIQUE

IV.1. OUTILS ET LANGAGE UTILISES

Afin de mener à bien ce projet, nous avons fait le choix de nombreux outils, langages et bibliothèques. Ils sont résumés dans le tableau ci-dessous :

Outil / Langage	Description	Raisons du choix
Java 21	Langage de programmation **orienté objet**, **portable** grâce à la machine virtuelle	C'est le langage avec lequel l'application de l'entreprise était développée, il est sécurisé, il intègre HttpClient et Xchart pour les graphiques
Swing	Bibliothèque UI desktop pour java	Offline, déploiement simple (pas de serveur), maîtrise rapide
Maven	Outil de build et gestion de dépendances	Plugins (tests, couverture, assembly), reproductible
JUnit 5 + Mockito	Framework de tests unitaires et mocks	Tests clairs, mocks faciles (PVGISClient), exécution headless stable

JaCoCo	Couverture de code	Mesure ciblée sur le métier, pilotage qualité par métriques
Java HttpClient	Client HTTP (Java 11+)	TLS natif, timeouts/retry, API moderne sans dépendance externe
Git / GitHub	Gestion de versions, collaboration	Traçabilité, reviews, CI possible
Windows 10/11	OS cible d'exécution	C'est le seul OS utilisé en entreprise

Tableau 2 : Outils utilisés

IV.2. PRESENTATION DES FONCTIONNALITES : DIAGRAMMES DE SEQUENCES PERTINENTS

Un diagramme de séquence technique est une vue dynamique d'un cas d'utilisation qui ordonne dans le temps les messages entre participants (UI, contrôleurs, services…), afin de clarifier les interactions d'exécution/communication entre les different.e.s classes/objets du système.

Afin de présenter en détails l'implémentation de nos fonctionnalités nous avons décidé de mettre en avant les diagrammes de séquence technique de trois principaux cas d'utilisation : Estimer la production, Voir les graphes financiers, Exporter en PDF/CSV.

Néanmoins, par souci de lisibilité, les DTO/modèles ne sont pas représentés exhaustivement dans les diagrammes (pas de lifelines ni d'attributs détaillés). Et plus de détails sur les classes sont donnés dans le diagramme de classe technique (*voir annexe 5*) et dans la java Doc très fournie de l'application.

IV.2.1. Cas d'utilisation : Estimer la production

L'objectif de ce cas d'utilisation est de faire un appel à l'API de PVGIS avec les bons paramètres d'URL, de Parser et utiliser les données de la réponse d'API pour tracer des graphes. Voici une description textuelle de ce cas d'utilisation :

Préconditions	• Écran "Estimation Grid" ouvert, les boutons, ''Export'' et ''Voir graphes'' ne sont pas cliquables et le bouton ''Finance'' est invisible. • Champs requis sont saisis et sont dans les bornes définies : latitude, longitude, puissance, inclinaison, azimut, pertes. • Connexion Internet disponible (accès API PVGIS).
Scénario nominal	• L'utilisateur clique "Estimation de production". • Le contrôleur valide les entrées puis lance un traitement asynchrone (SwingWorker). • Le service PVGIS appelle l'API, parse le JSON, mappe vers les modèles et agrège (mensuel → annuel). • L'UI est mise à jour : statut OK, graphes Xchart et tableaux alimentés ; les boutons d'actions réactivées.

	• Il clique sur le bouton ‘’Voir graphe’’ qui est désormais actif • Les graphes de production annuelles d’énergie s’affichent
Postconditions	• Résultats visibles (mensuel + annuel), cohérents et normalisés. • Le bouton du volet Finances et le bouton des Exports (CSV, PDF) sont visibles et opérationnels.
Scénarios alternatifs	• Entrées invalides : validation échoue → message explicite, champs mis en évidence, aucun appel API. • Erreur réseau/API (timeout, proxy, indisponible) : statut rouge, actions pertinentes désactivées, message utilisateur ; possibilité de “Réessayer”. • Réponse partielle/inattendue : champs optionnels manquants tolérés (valeurs par défaut documentées) ou rejet contrôlé si incohérences (ex. mois hors 1…12).

Tableau 3 : Description textuelle - Estimer la production

Nous avons fait le diagramme de séquence du cas d’usage « estimation de production » uniquement pour les systèmes Photovoltaïques couplés au réseau (« Grid »). En effet, on a des diagrammes similaires pour les deux autres systèmes PV (hors réseau et suiveur). Voici le diagramme :

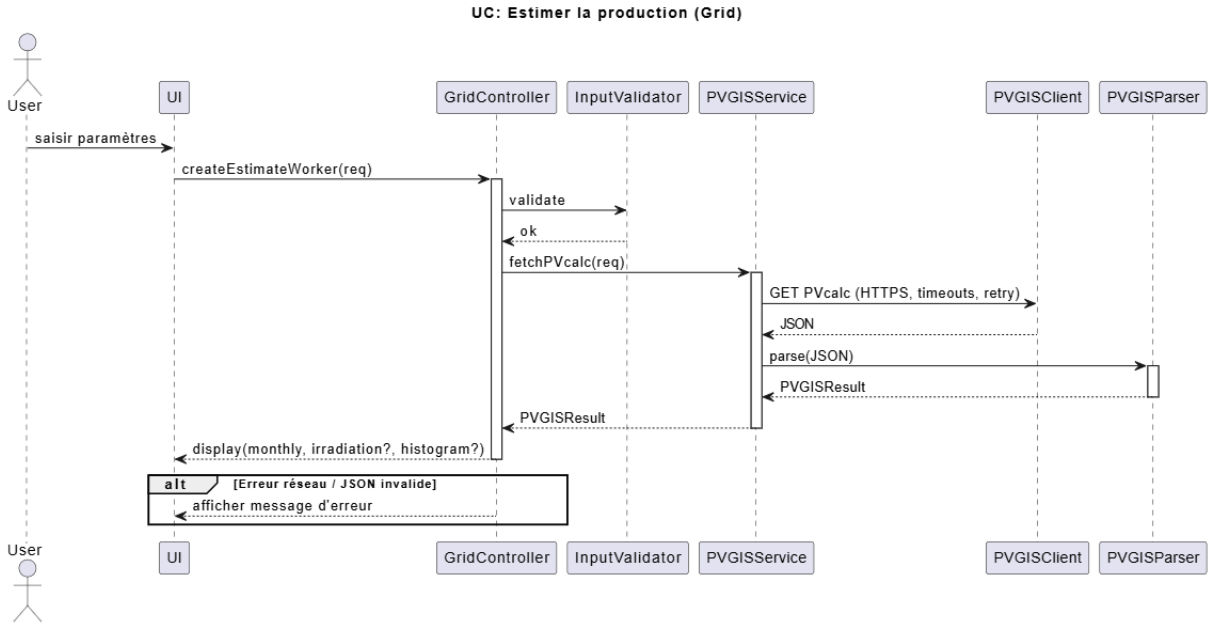


Figure 1 : diagramme de séquence - "Estimer la production"

Il est important de souligner quelques détails d’implémentation pertinents pour ce cas d’utilisation :

- ❖ Asynchronisme & UX : On utilise SwingWorker pour l’appel réseau et le parsing json de la réponse. SwingWorker est une classe Swing faite pour exécuter des tâches longues en arrière-plan (sur un autre thread) sans bloquer l’UI.

- Implémentation de graphes d'analyse : Ils ont été faits avec la bibliothèque java Xchart. Ces représentations aident l'utilisateur à analyser et visualiser les données issues de l'API PVGIS. Nous avons implémenté 6 graphes différents :
    - La production mensuelle d'énergie, l'Energie perdue mensuelle,
- Tolérance au schéma JSON PVGIS : certains champs dans le message envoyé à l'API de PVGIS (par exemple les champs histogramme et irradiation) peuvent manquer, alors on ajoute des valeurs par défaut pour éviter les incohérences et les crashs.
- Résilience réseau : le format HTTPS est obligatoire ; on a des connectTimeout, des requestTimeout, 2 retries (backoff) et on relance l'appel d'API s'il y a une InterruptedException (pas d'absorption silencieuse).
- Validation d'entrée en amont : On a des bornes de validation strictes (lat./lon, pertes 0–100, angle 0–90, azimut −180–360, durée 1–60, ratio 0–1) afin d'avoir messages clairs avant tout appel réseau.

IV.2.2. Cas d'utilisation : Voir les graphes financiers

L'objectif de ce cas d'utilisation est de calculer et visualiser les données de rentabilité financières de notre système pv à partir des données d'estimations reçues de PVGIS. Voici sa description textuelle :

Préconditions	• L'utilisateur a déjà une estimation de production via l'API PVGIS • On a cliqué sur « Finance » et on a saisi les paramètres financiers requis
Scénario nominal	• L'utilisateur clique sur « Tracer les graphes » • Le contrôleur récupère la production et les paramètres financiers. • Le service Finance effectue les calculs • Génération des séries de valeurs pour tracer les graphes (ex. : Cashflow annuel, Cashflow cumulé, VAN par année, …). • Le service Finance passe les séries à Xchart pour créer des graphes • Le contrôleur met à jour l'UI : panneaux de graphes remplis.
Postconditions	• Graphes financiers visibles e • Possibilité d'export (PDF/CSV) de rapport avec les valeurs financières calculées.
Scénarios alternatifs	• Paramètres financiers invalides (ex. : durée = 0, taux négatif, liste coûts plus courte que la durée) : la validation échoue → message clair, aucun graphe généré, anciens (s'ils existaient) peuvent être grisés.

Tableau 4 : Description textuelle - Voir graphes financiers

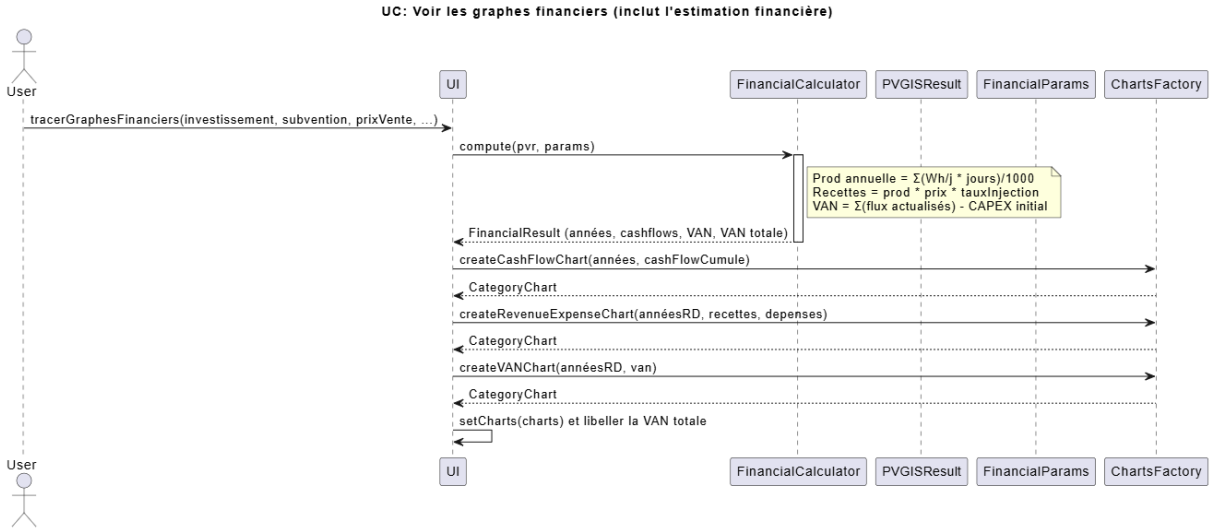


Figure 2 : diagramme de séquence - "Voir graphes financiers"

Voici des remarques pertinentes pour ce cas d'usage :

- De base, l'entreprise voulait juste les fonctionnalités d'estimation de production, mais c'est de ma propre initiative que j'ai proposé d'intégrer ce volet financier aux estimations pour centraliser encore plus les étapes d'étude photovoltaïque d'un client. Ainsi, J'ai décidé de tracer 3 graphes financiers présents dans AutoCalSol. Ils permettent d'avoir un aperçu de la rentabilité du système pv. Nous avons cherché les calculs sous-jacents de ces graphes d'AutoCalSol et avons adaptés les formules au contexte des photovoltaïques avec mon maitre de stage.
  - **Graphe 1 – Recettes(revenus) vs Dépenses (par année) :** Le graphe affiche typiquement (pour chaque année) deux barres : Recettes et Dépenses.

Formules	Définition des abréviations
❖ (Année $t \geq 1$) : Recettes : $Rev_t = Éco_t + RecInj_t$ Dépenses : $Dep_t = O\&M_t$ Cash-flow [6] : $CF_t = Rev_t - Dep_t$ $Éco_t = E\_auto_t \times Tariff\_auto$ $RecInj_t = E\_inj_t \times Tariff\_inj$ ❖ Année 0 (investissement initial) : Rev_0 = Subv (si subvention) ; Dep_0 = Invest ; $CF_0 = -$ Invest + Subv	Invest : coût initial total du projet. Subv : subvention encaissée au départ. $Éco_t$: économies (kWh autoconsommés × prix d'achat évité). RecInj : revenus de vente/injection (kWh injectés × tarif de rachat). $O\&M_t$: coûts annuels d'exploitation/maintenance. Rev_t : total des recettes annuelles. Dep_t : total des dépenses annuelles. CF_t : flux net ($Rev_t - Dep_t$).

Tableau 5 : Formules des recettes Vs dépenses

  - **Graphe 2 - Cash-flow Cumulé :** Ce graphe montre quand l'investissement est "remboursé" (payback simple).

Formules	Définition des abréviations

$CUM_0 = CF_0$ $CUM_t = CUM_{t-1} + CF_t$	CUM_t : somme des CF depuis l'origine. Payback simple : année où CUM_t devient ≥ 0.

Tableau 6 : Formules cash-flow cumulé

- **Graphe 3 - VAN (valeur actualisée nette) cumulée / NPV Over Time [7] :** Ce graphe montre la création progressive de valeur "temps ajusté" et permet de voir à partir de quelle année le projet devient réellement créateur de valeur au taux d'actualisation choisi.

Formules	Définition des abréviations
$DCF_t = CF_t / (1 + r)^t$ $VAN_0 = CF_0$ $VAN_t = VAN_{t-1} + DCF_t$ VAN finale = $\Sigma (CF_t / (1 + r)^t)$ (t = 0…n)	r : taux d'actualisation choisi. DCF_t : CF_t actualisé (valeur temps). VAN_t : valeur actualisée nette cumulée jusqu'à t. Payback : année où CUM_t devient ≥ 0. Horizon n : nombre d'années étudiées.

Tableau 7 : Formules VAN

- Gestion des cas extrêmes : par exemple, si dans le formulaire, la variable Durée = 0 (projection financière sur zéro année) alors on retourne des listes de valeurs pour graphes vides et on n'exécute pas les fonctions d'affichage de graphe.
- Séparation présentation/calcul : On a du calcul pur en service (sans UI), graphes produits ensuite à partir des séries pour testabilité.
- Performances : On ne régénère pas les données permettant de tracer les graphes si les paramètres n'ont pas changé. Lors de chaque export de rapport, on utilise juste les données stockées dans un objet de type « FianancialResult » sans refaire les calculs à chaque fois.

IV.2.3. Cas d'utilisation : Exporter en PDF/CSV

L'objectif de ce cas d'utilisation est de d'extraire un rapport en pdf ou csv contenant les tableaux des données ayant permis de tracer tous les graphes (d'estimation de production et financiers) ainsi que les graphes eux-mêmes. Voici une description textuelle de ce cas d'utilisation :

Préconditions	• Une estimation (production) et, si choisi, des calculs financiers sont déjà réalisés (données présentes en mémoire). • L'utilisateur a cliqué sur "Export" et a choisi le format (PDF/CSV). • L'utilisateur a entré un nom de fichier valide • Le chemin du dossier de destination est accessible en écriture
Scénario nominal	• L'utilisateur clique "Exporter" et choisit le format (PDF ou CSV). • Le contrôleur fait appel aux services adéquats pour construire le pdf / le fichier csv • Le fichier est ensuite écrit sur le système de fichiers. • Le contrôleur informe l'UI (message de succès).
Postconditions	• Fichier PDF ou CSV présent sur le disque et lisible. • UI à nouveau interactive (boutons réactivés).
Scénarios alternatifs	• Annulation utilisateur (dialogue fermé) : aucun fichier n'est créé, message "Export annulé". • Destination non accessible (droits, disque plein, chemin invalide), on capture une exception "Impossible d'écrire le fichier : …" ;

Tableau 8 : Description textuelle - Exporter en PDF/CSV

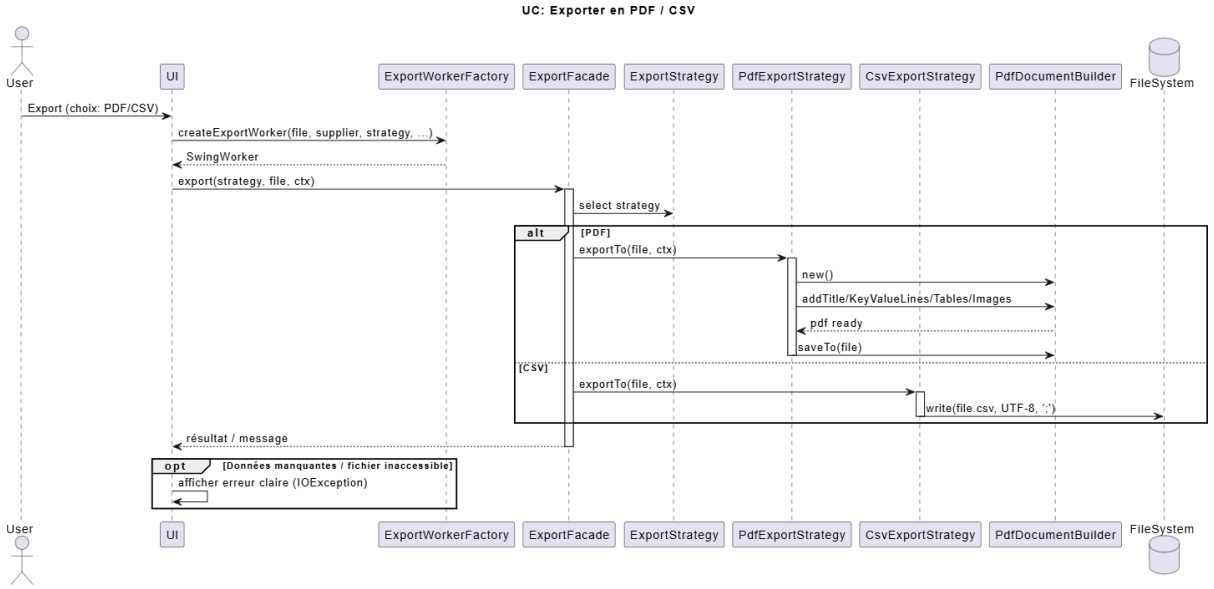


Figure 3: diagramme de séquence - "Exporter en PDF/CSV"

Comme pour les deux cas d'utilisations précédents, voici quelques particularités d'implémentation que nous voulons soulever pour l'export de rapport :

- Mise en page uniforme pour tous les types de PV avec une pagination automatique au format "Page X/Y".
- Gestion des Colonnes optionnelles : la colonne « Irradiation » est affichée uniquement si elle est présente et alignée en taille ; sinon on masque la colonne.
- Sécurité : Assainir le nom de fichier (pas d'../, pas de caractères interdits).
- Erreurs maîtrisées : un contexte incomplet déclenche un IOException claire et on gère les accès refusés/chemins invalides.
- Asynchronisme : l'export se fait en arrière-plan avec SwingWorker.

IV.2. PRESENTATION DES INTERFACES

Pour ce qui est des vues, j'ai créé plusieurs pages et ajouté des boutons et menus pour améliorer l'ergonomie de l'application interne de l'entreprise. D'abord, j'ai créé une page d'accueil et une page transiente d'estimation de production comportant les différents types de systèmes pv gérés par notre application. L'application n'avait pas vraiment de page d'accueil, elle s'ouvrait directement sur un formulaire pour les pré-études photovoltaïques, sa création fut une de mes initiatives. Elle comporte un bouton « Tutoriel » qui ouvre une boite de dialogue qui explique comment naviguer entre les pages. Ensuite, J'ai créé les trois pages d'estimation pour les 3 différents types de pv (comportant formulaire, menus et conteneur de graphes). Aussi, il faut noter que toutes ces pages sont parfaitement responsives. Enfin, j'ai créé une barre de navigation pour toutes les pages de l'application contenant un bouton « Accueil », un bouton « Estimer Production » et un bouton « Pré-étude » afin d'accéder de manière efficiente à toutes les pages de l'application.

Remarque : la charte de couleurs (bleu, vert, jaune) a été choisie en fonction des couleurs du logo de l'entreprise et validées par le maitre de stage.

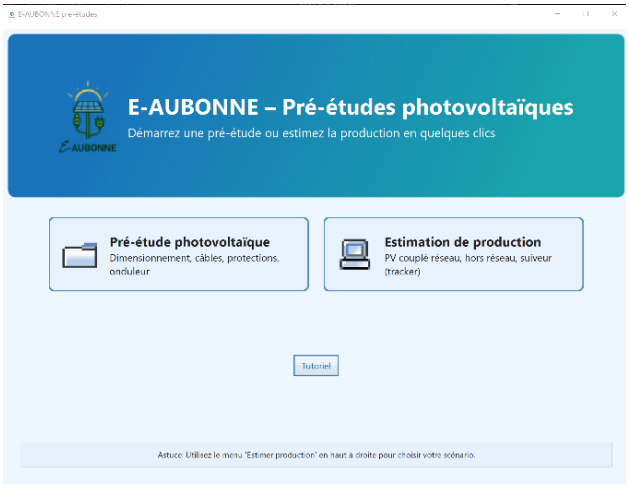


Figure 5 : page d'accueil

Figure 4 : page des options d'estimation

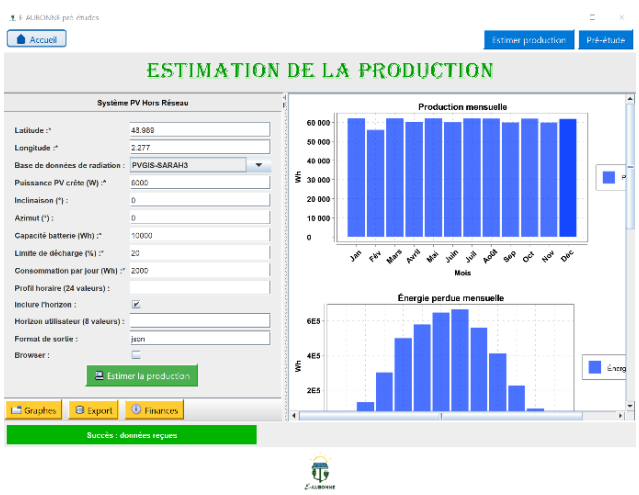


Figure 7 : Page d'estimation pv Grid

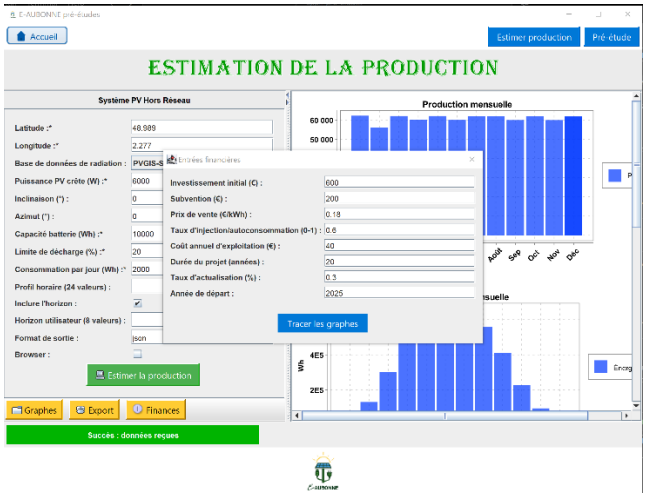


Figure 6 : Page du formulaire financier

CHAPITRE V : TEST ET METRIQUE D'EVALUATION DE LA SOLUTION

V.1. STRATÉGIE DE TEST

Afin d'effectuer nos tests, nous avons utilisé comme données des réponses JSON de l'API PVGIS stockées dans le projet (afin que les tests unitaires n'aient pas besoin de réseau pour s'exécuter) et des inputs financiers dont nous maitrisons les résultats. De plus, tous les tests effectués sont headless (exécution sans affichage). Nous avons divisé nos 71 tests en 3 grandes familles de test :

- Les Tests unitaires (52) concernent principalement : le Parser PVGIS (variantes de schéma, champs manquants, conversions E_m→Wh/j) ; le service finance (séries, VAN, bornes, arrondis) ; l'export (CSV lignes/encodage/points-virgules, …) et la validation d'entrée (bornes, types).

- Les Tests d'intégration (11) légers pour la classe PVGISService avec client HTTP mock (timeouts, retry) et les Contrôleurs : SwingWorker happy-path et erreurs (callbacks EDT).
- Tests UI/End-to-end (8) : Sans ouvrir de fenêtres(headless), on vérifie que les callbacks mettent à jour l'état (labels/flags).

Pour savoir comment exécuter les tests et pour avoir une trace d'exécution des tests, *voir annexe 2*.

V.2. QUELQUES CAS DE TEST

- Test de PVGISParser : JSON partiel + normalisation des unités pour vérifier robustesse aux données réelles/partielles de l'API.
  - Entrées : JSON sans irradiation (et histogramme optionnel), valeurs en Wh.
  - Vérifie : pas d'exception, valeurs par défaut documentées, unités normalisées (Wh→kWh), structure de sortie cohérente.
- Test du Contrôleur Grid (PV couplé au réseau) : gestion d'erreur côté UX et la résilience.
  - Entrées : échec mocké du service (PVGIS/validation).
  - Vérifie : statut d'erreur (rouge), actions/toolbar désactivées, message utilisateur clair ; en succès, graphsPanel peuplé et toolbar activée.

V.3. MÉTRIQUES D'ÉVALUATION DU CODE

- Couverture de test (JaCoCo) : code couvert à 84% en ciblant les packages dans lesquels nous avons travaillé (*voir annexe 3*).
- Robustesse : Ratio de fixtures JSON parsées avec succès (objectif : 100% du set) ; Absence d'exceptions en export (N exécutions).
- Performance : parsing + calcul + export < 2s sur poste dev ; taille PDF/CSV raisonnable (une centaine de ko pour le PDF et moins de 5ko pour le csv).
- Qualité code : pour cette partie nous avons utilisé l'outils d'analyse de code statique SonarQube et nous avons les résultats (*voir annexe 1*) suivants :
  - Maintenabilité et évolutivité du code = score A ce qui signifie que la **dette technique** (le temps estimé pour corriger les problèmes de style, de complexité, de duplication, etc.,) est négligeable et que le code est propre et facile à comprendre
  - Sécurité du code = Score A ce qui signifie que mon code est sûr, respecte les standards de sécurité de base et qu'il ne possède pas de vulnérabilités connues.
  - Fiabilité = Score D car il y'a des soucis de convention de nommage dans des classes que je ne suis pas autorisé à toucher.

PARTIE 3 : BILAN GENERAL DU STAGE

CHAPITRE VI : PRESENTATION ET ANALYSE DES RESULTATS

VI.1. ATTEINTE DES OBJECTIFS PAR RAPPORT AUX OBJECTIF ET METHODES D'EVALUATION DE CES RESULTATS

Nous avons regroupé les objectifs initiaux, les résultats obtenus et les méthodes d'évaluation de ces résultats dans ce tableau :

Objectif (attente)	**Résultats (atteints / attendus)**	**Méthodes d'évaluation**
Concevoir, intégrer et analyser les données d'estimation de production (Grid, Off-Grid, Tracker)	- Implémentation d'un module d'estimation des données de production d'énergie des 3 types de systèmes photovoltaïques. Ce module est basé sur des appels à l'API PVGIS. Les données (Json) d'estimations reçues de l'API sont parsées et encapsulées en prenant en compte les réponses d'API complètes ou partielles. - Création exacte des graphes d'analyse de données de PVGIS.	- Vérification croisée : L'entreprise nous a fourni un jeu de données d'un ancien client pour effectuer les tests. Ces tests nous ont permis de conclure que les graphes et données d'estimation issues de notre application finale sont identiques à ceux que l'on obtient en faisant des estimations directement sur le site web PVGIS ou sur AutoCalSol. - Jeux de tests Parser (JSON complet/partiel, valeurs limites)
Réduire la complexité de préparation d'un rapport d'estimation ainsi que les risques d'erreurs manuelles dus à l'usage d'outils multiples pour l'estimation (PVGIS et AutoCalSol si le système pv est de type suiveur ou de type hors réseau)	- Génération automatique de rapports PDF/CSV incluant les tableaux et les graphes associés aux données de production et aux indicateurs financiers(si disponibles) - Réduction du nombre d'étapes pour créer un rapport client : on passe de au plus 5 étapes (voir section II.2.2.2 Processus actuel) à exactement 2 étapes (remplissage des données dans les formulaires de l'application et génération de	- Feedback utilisateur (voir les appréciations du président de l'entreprise dans la fiche de stage) - Comptage étapes manuelles éliminées (voir section II.2.2.2 Processus actuel)

Objectif (attente)	Résultats (atteints / attendus)	Méthodes d'évaluation
	rapport) - Centralisation des calculs et suppressions des ressaisies	
Modéliser les données et calculs puis implémenter les indicateurs financiers (VAN, etc.)	Implémentation de FinancialCalculator (flux annuel, VAN, TRI, payback…)	- Tests unitaires sur formules (valeurs connues / cas limites) - Comparaison avec des données de référence de l'entreprise
Créer une interface ergonomique, valider les entrées et orchestrer via des contrôleurs fiables	- UI réactive (Swing + SwingWorker) ; - Amélioration considérable de la beauté visuelle de l'application (plus moderne) ; - Création de nouvelle page pour l'ergonomie (création d'une page d'accueil, ajout des boutons et menus pour faciliter la navigation entres pages) --- Inhibition des boutons en cas erreur.	- Tests contrôleurs (succès / échec mocké)
Industrialiser la qualité : tests unitaires, couverture ciblée, analyse statique, documentation d'usage	- Suite de tests (Parser, finance, export, contrôleurs); - Profil JaCoCo (code de mes fonctionnalités couvert à 84%) ; - SonarQube: notes A (maintenabilité et sécurité), note D en fiabilité; - Documentation : README, Tutoriel, Java Doc, diagrammes UML(composants, classes techniques, séquences techniques) + description textuelle des cas d'utilisation rédigée.	- Rapport JaCoCo (voir capture d'écran en annexe) - Tableau de bord SonarQube (voir capture d'écran en annexe) - Revue documentaire (voir en annexe)

Tableau 9 : Objectifs, résultats et méthodes d'évaluation

VI.2. CRITIQUES DES RESULTATS

Bien que les objectifs fonctionnels principaux aient été atteints, certaines limites demeurent : Premièrement, la validation des estimations de notre application ne se fait que sur un seul jeu de données fourni par l'entreprise. Il faudrait plusieurs jeux de données divers en plus pour garantir que la solution est excellente. Deuxièmement, on note l'absence de métriques quantitatives sur le gain de temps utilisateur. On ne se base que sur les feedbacks de l'utilisateur. Troisièmement, nous n'avons pas géré certaines hypothèses financières (par exemple, avoir des recettes ou des dépenses qui varient chaque année). Aussi, notre solution est fortement liée à l'évolution de l'outils PVGIS. Enfin, il faudrait faire plus de tests pour couvrir tout le code.

CHAPITRE VII : BILAN PERSONNEL DU STAGE

VII.1. POINTS DURS MAJEURS ET SOLUTIONS

On a eu deux obstacles majeurs :

Intégrer mes fonctionnalités dans une application déjà existante, mal structurée et sur laquelle un autre stagiaire travaillait en parallèle. Solution : Structurer (avec l'accord de mon maitre de stage et de l'autre stagiaire) le projet en architecture MVC associés Services, Ajout des dépendances dont j'avais besoin dans le pom.xml (JaCoCo, …) créer un dépôt GitHub et des branches pour l'autre développeur et moi afin qu'on puisse avoir à chaque fois la version la plus récente du code.

Trouver comment intégrer les fonctionnalités d'estimation de l'application web PVGIS ainsi que les calculs de rentabilités financières d'AutoCalSol dans l'application Java. Solution : Usage de l'API de PVGIS (trouvé après quelques recherches) ; Recherche et utilisation des formules financières réelles derrières graphes tracés dans AutoCalSol

VII.2. COMPETENCES INTERPERSONNELLES ACQUISES EN ENTREPRISE

Durant ce stage, une autre qualité principale que j'ai développé pendant ce stage c'est la proactivité. En effet, j'avais pendant ce stage, la liberté de prendre de nombreuses initiatives soit dans l'optique de faciliter le travail de l'entreprise (par exemple l'ajout du module financier était une initiative) soit dans l'optique de rendre mon code le plus maintenable possible (Les diagrammes techniques, la couverture de tests et autres n'étaient pas demandés par l'entreprise, mais furent très appréciés). Aussi, j'ai travaillé sur mon adaptabilité dans un environnement dynamique. J'ai appris à m'intégrer dans un projet déjà existant, en travaillant main dans la main avec mes collaborateurs. Dans le meme sillage, j'ai appris à mieux m'organiser et me discipliner. J'ai compris que les outils de gestion de projet comme Jira ne permettaient non pas d'alourdir le travail mais de le maturer plus rapidement et plus efficacement. Toutes ces compétences et l'expérience immersive en entreprise me font me sentir enfin outillé pour affronter le métier d'ingénieur (ce qui n'était pas le cas à la fin de ma deuxième année).

CONCLUSION

Somme toute, l'application développée répond à la problématique initiale, en regroupant au sein de l'application interne d'étude photovoltaïque les fonctionnalités d'estimation de production (via l'API de PVGIS) et de rentabilité financière de tout type de photovoltaïques. Ainsi, nous avons produit une application fiable et reproductible (suite de 71 tests, code couvert à 84%, documentation techniques fournie avec la Java Doc, un Readme précis, des diagrammes UML, …) qui réduis de 3 étapes le processus d'étude photovoltaïque. Cette application inclue aussi des services d'export des rapports d'estimation en pdf/csv, le tout dans des interfaces chaleureuses et ergonomiques. Cependant, nous afin de compléter cette solution, nous pourrons ajouter des fonctionnalités de programmation d'alertes maintenance des systèmes pv, de facturation, … ou tout autre service que l'on peut proposer à un client dans le cadre d'une étude photovoltaïque.

REFERENCES BIBLIOGRAPHIQUES

[1] Description de E-Aubonne : https://e-aubonne.com/qui-sommes-nous/

[2] Définition Photovoltaïque : https://fr.wikipedia.org/wiki/%C3%89nergie_solaire_photovolta%C3%AFque

[3] Description de PVGIS : https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/getting-started-pvgis/pvgis-user-manual_en

[4] Documentation API PVGIS : https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/getting-started-pvgis/api-non-interactive-service_en

[5] Description d'AutoCalSol : https://autocalsol.ines-solaire.org/etude/production

[6] Notions de Cash-flow : https://www.investopedia.com/terms/c/cashflow.asp et https://www.compta-facile.com/cash-flow-definition-calcul-interet/

[7] Notion de VAN : https://www.investopedia.com/terms/n/npv.asp

ANNEXES

Annexe 1 : Résultats de l'analyse statique du code par SonarQube. On a un score A pour la sécurité, un score D pour la fiabilité et un score A pour la maintenabilité

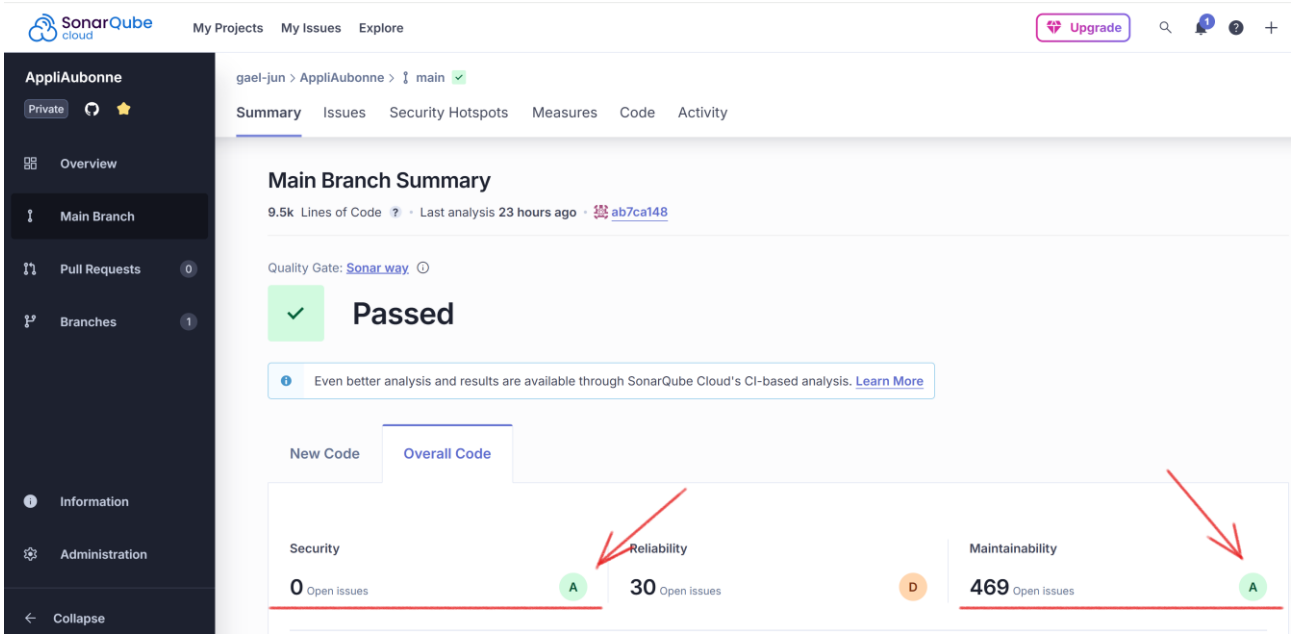


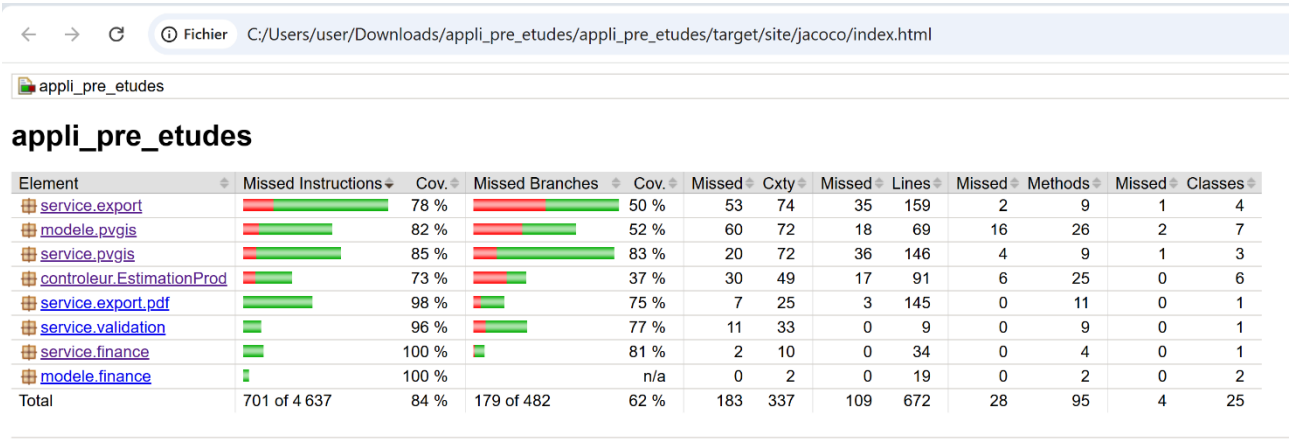
Figure 8 : Tableau de bord SonarQube

Annexe 2 : Trace d'exécution des 71 tests. On exécute les tests avec ''mvn test'' ou ''mvn verify -Pcoverage'' (génère aussi le rapport de couverture JaCoCo dans le chemin ''target/site/jacoco/index.html'')

```
[INFO] service.validation.InputValidatorTest.injectionRatioValid -- Time elapsed: 0.001 s
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 71, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ appli_pre_etudes ---
[INFO] Building jar: C:\Users\user\Downloads\appli_pre_etudes\appli_pre_etudes\target\appli_pre
_etudes-0.0.1-SNAPSHOT.jar
```

Figure 9 : Trace d'exécution des tests

Annexe 3 : Rapport JaCoCo de couverture du code des packages clé



Fichier C:/Users/user/Downloads/appli_pre_etudes/appli_pre_etudes/target/site/jacoco/index.html

appli_pre_etudes

appli_pre_etudes

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
service.export		78 %		50 %	53	74	35	159	2	9	1	4
modele.pvgis		82 %		52 %	60	72	18	69	16	26	2	7
service.pvgis		85 %		83 %	20	72	36	146	4	9	1	3
controleur.EstimationProd		73 %		37 %	30	49	17	91	6	25	0	6
service.export.pdf		98 %		75 %	7	25	3	145	0	11	0	1
service.validation		96 %		77 %	11	33	0	9	0	9	0	1
service.finance		100 %		81 %	2	10	0	34	0	4	0	1
modele.finance		100 %		n/a	0	2	0	19	0	2	0	2
Total	701 of 4 637	84 %	179 of 482	62 %	183	337	109	672	28	95	4	25

Figure 10 : Rapport JaCoCo

Annexe 4 : Diagramme de composants

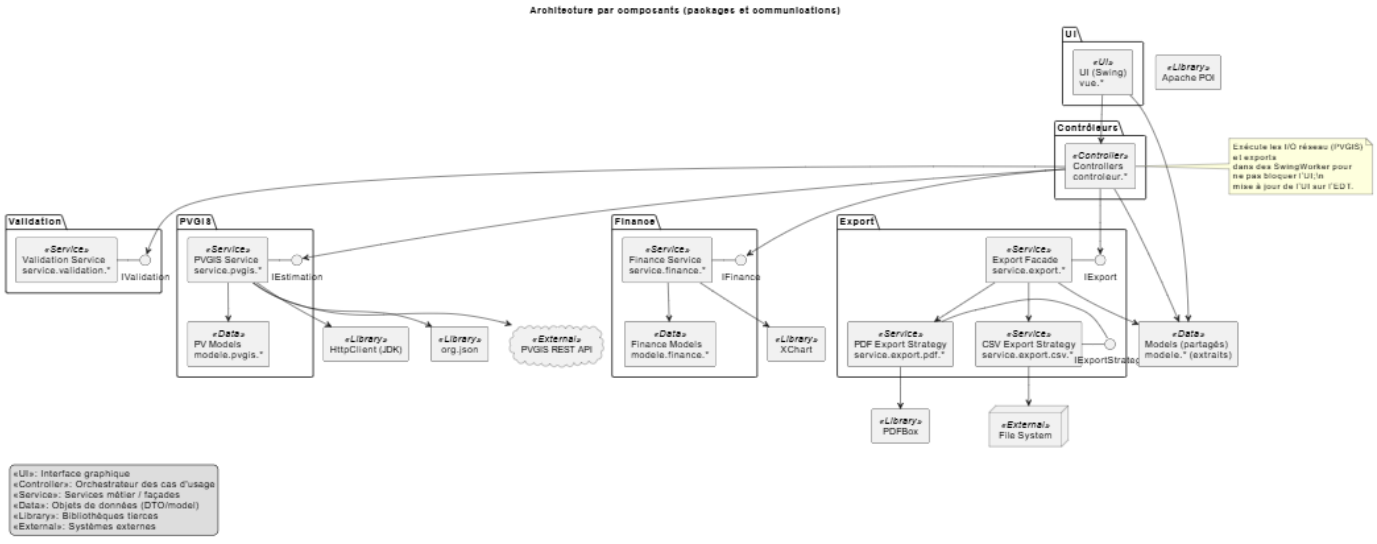


Figure 11 : Diagramme de composants

Annexe 5 : Diagrammes de classes techniques

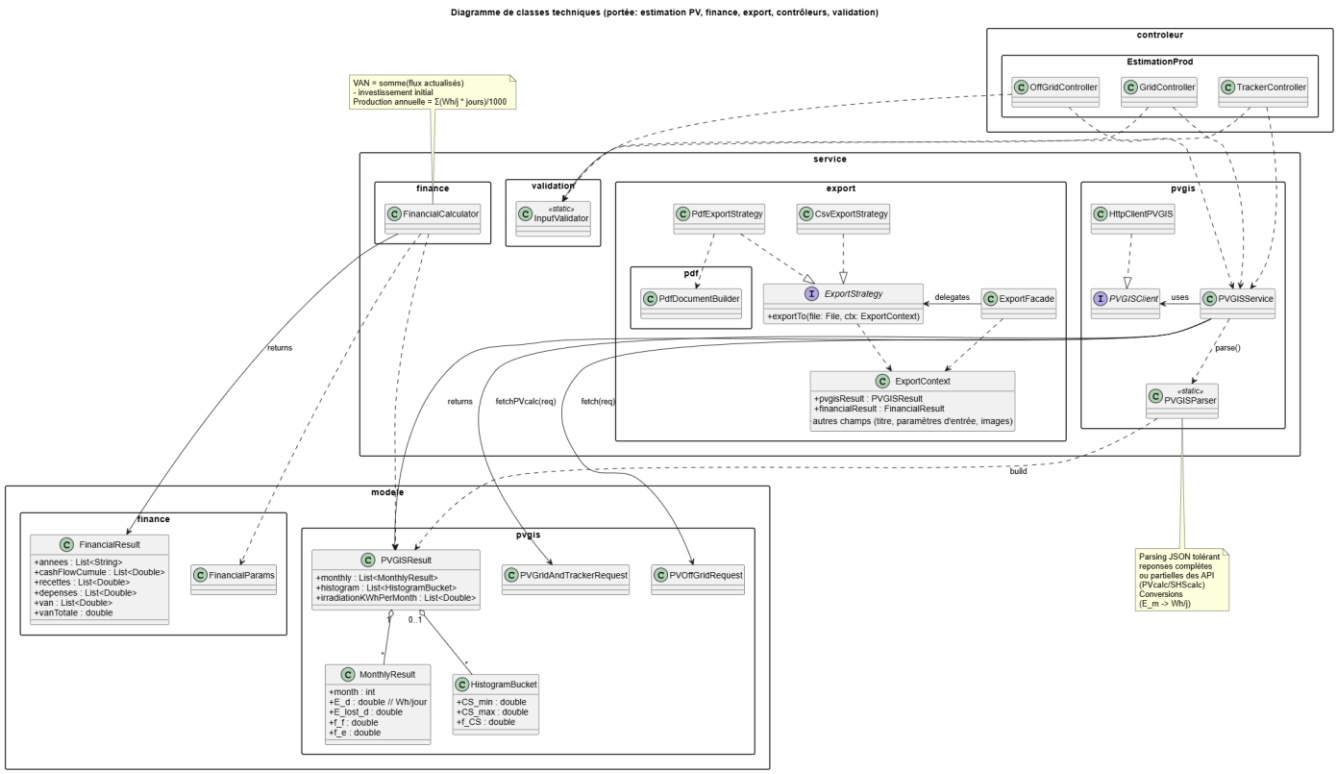


Figure 12 : Diagramme de classes techniques

Annexe 6 : Rapports PDF et CSV générés pour l'estimation de production et l'estimation financière d'un système PV couplé au réseau. Ils permettent d'avoir une vue d'ensemble sur les graphes que nous avons construit pour ce cas avec les tableaux des données associées.

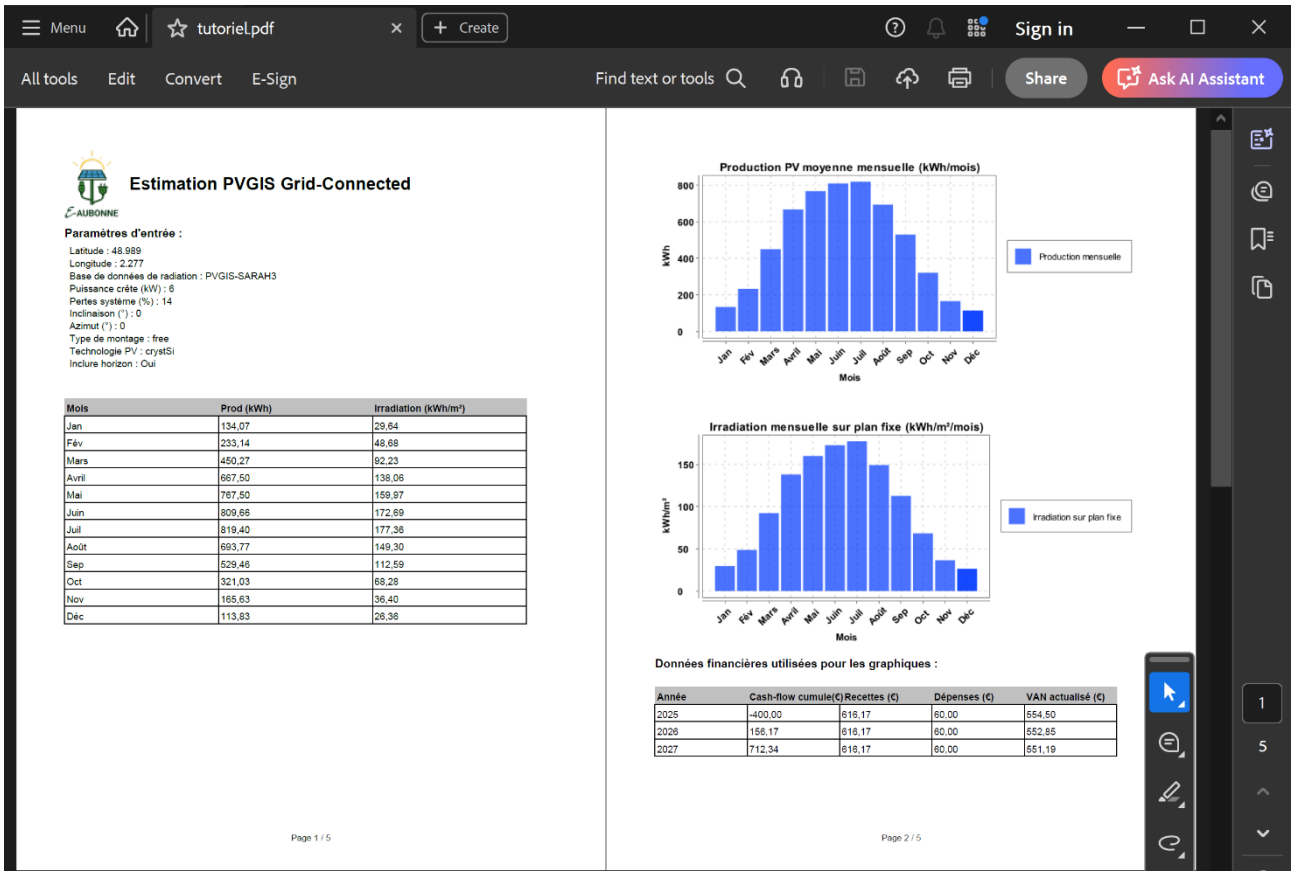


Figure 13 : Rapport pdf (1)

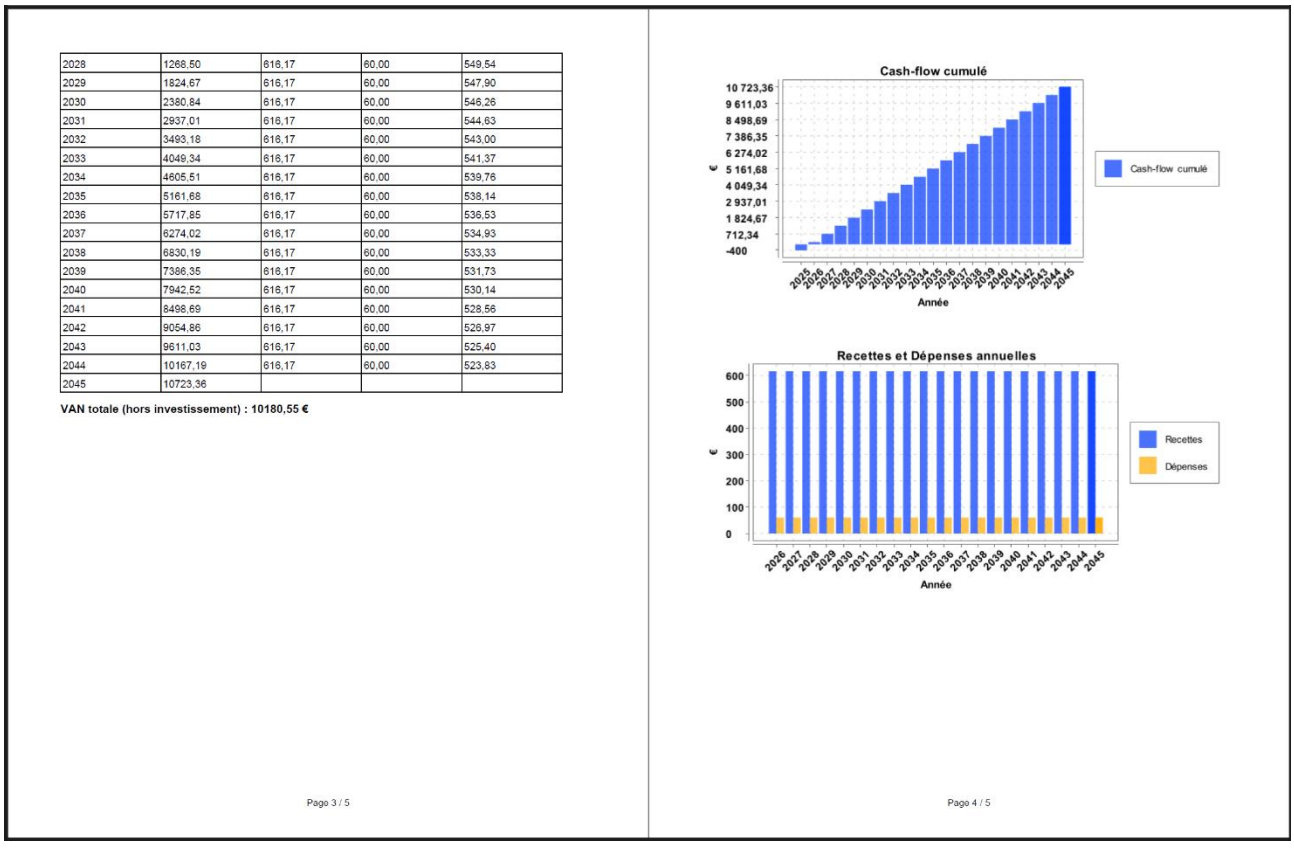


Figure 14 : Rapport pdf (2)

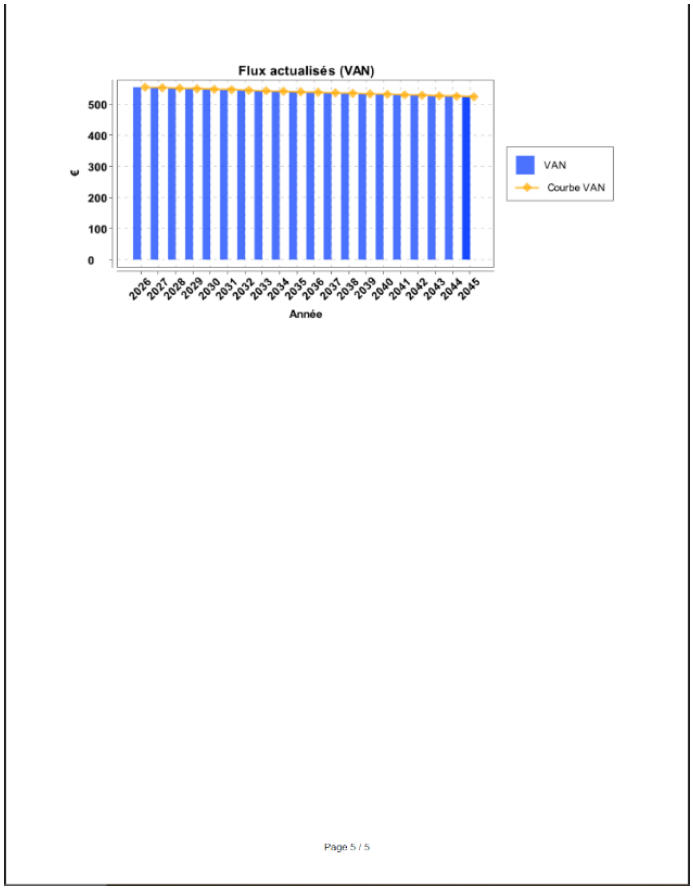
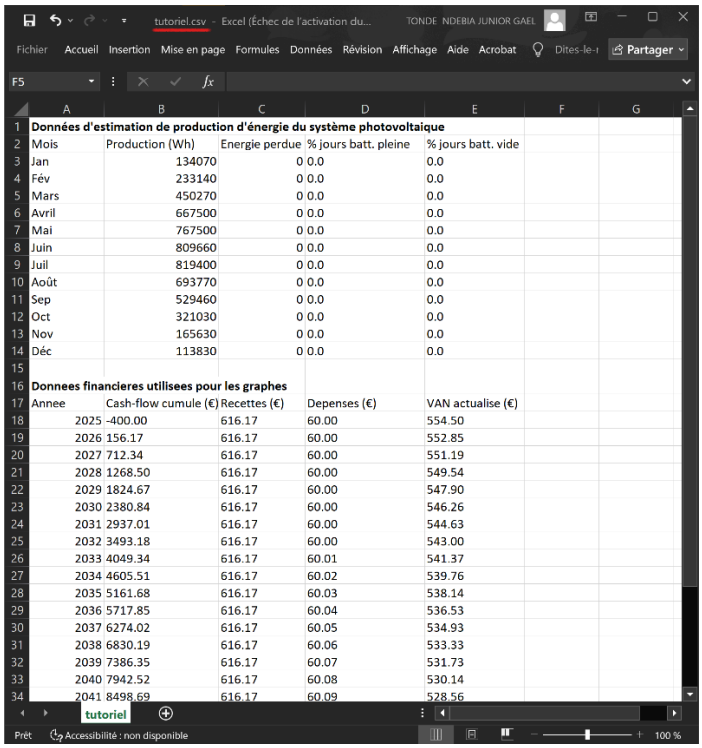


Figure 15 : Rapport pdf (3)



	A	B	C	D	E	F	G
1	**Données d'estimation de production d'énergie du système photovoltaique**						
2	Mois	Production (Wh)	Energie perdue	% jours batt. pleine	% jours batt. vide		
3	Jan	134070	0	0.0	0.0		
4	Fév	233140	0	0.0	0.0		
5	Mars	450270	0	0.0	0.0		
6	Avril	667500	0	0.0	0.0		
7	Mai	767500	0	0.0	0.0		
8	Juin	809660	0	0.0	0.0		
9	Juil	819400	0	0.0	0.0		
10	Août	693770	0	0.0	0.0		
11	Sep	529460	0	0.0	0.0		
12	Oct	321030	0	0.0	0.0		
13	Nov	165630	0	0.0	0.0		
14	Déc	113830	0	0.0	0.0		
15							
16	**Donnees financieres utilisees pour les graphes**						
17	Annee	Cash-flow cumule (€)	Recettes (€)	Depenses (€)	VAN actualise (€)		
18	2025	-400.00	616.17	60.00	554.50		
19	2026	156.17	616.17	60.00	552.85		
20	2027	712.34	616.17	60.00	551.19		
21	2028	1268.50	616.17	60.00	549.54		
22	2029	1824.67	616.17	60.00	547.90		
23	2030	2380.84	616.17	60.00	546.26		
24	2031	2937.01	616.17	60.00	544.63		
25	2032	3493.18	616.17	60.00	543.00		
26	2033	4049.34	616.17	60.01	541.37		
27	2034	4605.51	616.17	60.02	539.76		
28	2035	5161.68	616.17	60.03	538.14		
29	2036	5717.85	616.17	60.04	536.53		
30	2037	6274.02	616.17	60.05	534.93		
31	2038	6830.19	616.17	60.06	533.33		
32	2039	7386.35	616.17	60.07	531.73		
33	2040	7942.52	616.17	60.08	530.14		
34	2041	8498.69	616.17	60.09	528.56		

Figure 16 : Rapport csv

Annexe 7 : Manuel d'installation et d'utilisation (Un bout du Readme). Il permet de prendre en main l'application sans encombre par des futurs développeur. D'ailleurs, Il montre aussi comment accéder à la java doc très fournie que nous avons généré.

1) Prérequis et installation (Windows)

- Java 21 (JRE/JDK) : S'il n'est pas encore installé, le télécharger et l'installer via Microsoft Build of OpenJDK 21 à l'adresse https://learn.microsoft.com/java/openjdk/download

2) Lancer l'application à partir du JAR

Le build Maven produit un JAR "avec dépendances" prêt à l'emploi : `appli_pre_etudes-0.0.1-SNAPSHOT-jar-with-dependencies.jar`.

- Double-clic (si l'association .jar est configurée) OU en ligne de commande :

cd C:\Users\user\Downloads\appli_pre_etudes\appli_pre_etudes\target

java -jar appli_pre_etudes-0.0.1-SNAPSHOT-jar-with-dependencies.jar

3) Compiler et exécuter avec Maven (pour le développeur)

mvn clean package

java -jar target\appli_pre_etudes-0.0.1-SNAPSHOT-jar-with-dependencies.jar

5) Accéder à la Java Doc (API)

- Générer la Java Doc avec Maven puis y accéder via ce chemin target\site\apidocs\index.html :

mvn javadoc:javadoc