

5.6.1 repris par lena le 14/11/2018

5.4

## Explorer visuellement des données de simulation massives pour analyser le comportement d'un modèle.

Version 2018-09-21

### Sommaire

Introduction . . . . .	2
5.1 Capter les sorties de SimFeodal . . . . .	2
5.1.1 Masse des données . . . . .	2
5.1.2 RéPLICATIONS . . . . .	4
5.1.3 Expériences . . . . .	6
5.1.4 Des données aux indicateurs . . . . .	7
5.2 Comment explorer les données de SimFeodal ? . . . . .	9
5.2.1 Observation en direct vs a posteriori . . . . .	9
5.2.2 Générer les indicateurs . . . . .	12
5.2.3 Organiser les indicateurs en rapports paramétrables . .	14
5.2.4 Organiser les rapports : Dashboards . . . . .	17
5.2.5 Interagir avec les rapports : exploration interactive . .	19
5.2.6 Explorer en comparant : SimEDB . . . . .	23
5.3 Organiser les données . . . . .	26
5.3.1 Assurer la capacité d'interrogation des données . . . .	26
5.3.2 Structuration des données de SimFeodal . . . . .	37
5.4 Une plate-forme d'exploration de données de simulations : SimEDB . . . . .	43
5.4.1 Contraintes . . . . .	43
5.4.2 Construire une plate-forme interactive pour l'évaluation de SimFeodal . . . . .	49
Conclusion . . . . .	61

## 5.4 Une plate-forme d'exploration de données de simulations : SimEDB

La section 5.2 (Comment explorer les données de SimFeodal ?) a décrit les étapes successives d'avancement dans l'exploration des données en sortie de SimFeodal, depuis l'observation en direct des simulations jusqu'au besoin d'une plate-forme permettant l'exploration et la comparaison interactive des sorties de simulation. La plate-forme proposée en réponse à ce besoin, SimEDB<sup>29</sup>, dans un objectif de généricité et d'adéquation, se devait aussi de répondre à de nombreuses contraintes, aussi bien liées aux possibilités offertes qu'à l'usage qui en serait fait. Dans cette partie, nous nous attacherons donc à présenter les contraintes qui ont guidé la conception de SimEDB, ainsi que les choix, méthodologiques et techniques, qui en ont résulté.

au est "à la volée"?

### 5.4.1 Contraintes

#### 5.4.1.1 Adapter la complexité aux utilisateurs

Dans le domaine de l'Interface Homme-Machine (IHM), il est courant de considérer qu'un outil d'analyse et de représentation doit être adapté à un public. La figure 5.11, emblématique de la conception de géovisualisations par Alan MacEachren, replace ainsi les types d'usage d'une plate-forme d'exploration selon trois axes : les utilisateurs visés (*users*), le niveau d'interaction souhaité (*interaction*) et l'objectif poursuivi par la (géo)visualisation (*task*). D'après ce schéma, à un niveau d'expertise de l'utilisateur correspond un unique degré d'interaction : plus l'utilisateur est expert du domaine, plus il s'attendra à disposer d'un outil complexe : « All participants agreed that user expertise requires increased interface complexity, as suggested by the Cartography<sup>3</sup> framework » (ROTH 2015, p. 16).

res. form. lektör  
l'atén.

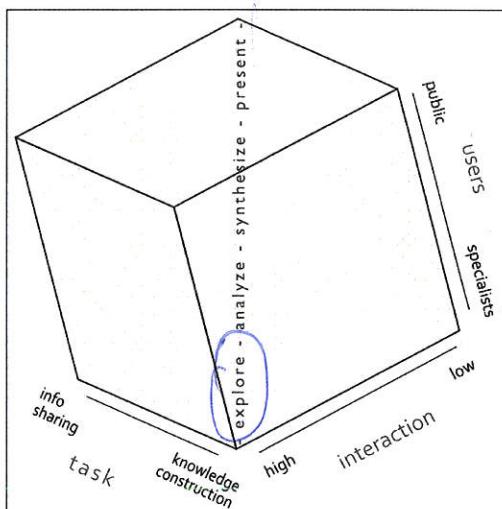


FIGURE 5.11 – « An update to Cartography<sup>3</sup>, 10 years after its conception », par CÖLTEKIN, JANETZKO et FABRIKANT 2018, d'après MAC EACHREN et al. 2004, p. 10.

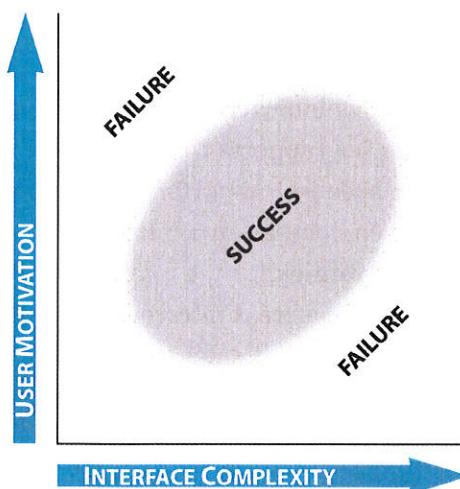


FIGURE 5.12 – « Interface complexity versus user motivation. », ROTH 2013, p. 79.

L'usage prévu de SimEDB – nettement dans la construction de connaissance

29. SimFeodal Exploration Dashboard, voir la note de bas de page 15, page 25.

–, de même que le « niveau » de ses utilisateurs – expert dans le sens thématique et de la modélisation conceptuelle –, devraient donc logiquement, selon le schéma, bénéficier d'une forte complexité possible dans l'interaction.

**Des utilisateurs hétérogènes mais captifs** SimEDB s'inscrit pourtant à un niveau intermédiaire, entre l'analyse et la synthèse : l'exploration, au sens entendu par MacEachren (*explore*), est ici délaissée de par le choix de présenter des indicateurs spécifiés plutôt que de laisser l'utilisateur les assembler.

Cet écart au modèle conceptuel de MacEachren s'explique notamment par la diversité des utilisateurs de SimEDB : qualifier un niveau d'expertise général serait absurde, tant les spécificités de cette expertise sont nombreuses : entre des profils de spécialiste thématiciens, de modélisateurs ou encore de géomaticiens, l'expertise est présente, mais concernant des champs différents, toutefois tous intéressés par l'exploration des sorties de SimFeodal.

Il est dès lors peu évident de se fixer sur un degré de complexité à atteindre dans la plate-forme d'exploration : un niveau faible serait frustrant pour les utilisateurs avancés, et un niveau avancé serait source de confusion et donc de perte de motivation. *pour*

Il nous était toutefois possible de miser sur une bonne motivation générale étant donné les circonstances particulières d'utilisation de SimEDB : contrairement à une utilisation grand public, qui ne présente aucun engagement vis-à-vis d'une interface d'exploration de données, ou à l'inverse contrairement à des domaines experts où chaque utilisateur dispose de ses propres outils et méthodes pour explorer un jeu de données, le public cible de SimEDB est « captif », c'est-à-dire qu'il ne dispose pas d'autre solution que de passer par cette plate-forme pour explorer les données, en particulier en raison des contraintes liées à ces données (cf. section 5.1.4, Des données aux indicateurs par exemple).

Dès lors, la motivation des utilisateurs ne peut qu'être importante, et l'interface a la possibilité de devenir plus complexe tout en se révélant adaptée (figure 5.12).

**Intuitivitivé de l'usage au regard des applications traditionnelles** En dépit de cette motivation, les utilisateurs de SimEDB demeurent majoritairement des experts thématiciens, potentiellement peu familiarisés à l'exploration de données interactives. Afin que le temps d'exploration des données issues de SimFeodal soit dévolué à la compréhension et à la synthèse de ces données plutôt qu'à un apprentissage ou amélioration en exploration de données, il a été choisi de créer une application aussi simple que possible au regard des fonctionnalités principales qu'elle devait permettre : observer les indicateurs de sortie de simulation pour des expériences données, et les comparer entre elles aussi efficacement que possible. Il n'était donc pas question de construire un nouveau « logiciel expert », doté de dizaines de fonctionnalités avancées, mais au contraire, de simplifier au maximum l'interface pour ne pas encombrer et complexifier l'utilisation de ces fonctionnalités principales. On souhaitait une plate-forme aussi épurée que possible, plutôt que de partir, par exemple, sur la personnalisation et l'adaptation de l'un des outils d'exploration existants et dédiés à offrir une forte possibilité de manipulation.

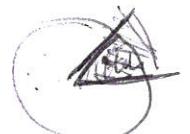
#### 5.4.1.2 Efficacité

Dans la description du choix du SGBD, on a mentionné une première fois l'intérêt de disposer d'une solution d'interrogation de données permettant une rapidité de l'exécution des requêtes. Sans entrer dans le détail des recherches en

?  
expliqué ?  
Rendre

plus  
résultats de  
simulation ?

) iv. dire pourquoi  
recherché à faire  
dans un niveau bien  
spécifique ?



IHM, on peut toutefois caractériser cet intérêt par deux aspects complémentaires, une solution interactive minimisant les latences permettant (1) de conserver l'utilisateur, c'est-à-dire de ne pas le décourager d'utiliser l'application, et (2) de lui faire conserver sa concentration (*focus*), c'est-à-dire que le délai entre son interaction et la réponse graphique soit suffisamment court pour que l'identification effectuée par son système cognitif ne soit pas affecté.

Le premier point a été abordé plus haut (section 5.3.1.2), et surtout, en raison de la « captivité » de l'utilisateur évoquée ci-dessus, ne s'applique que marginalement à notre cas d'étude : si des délais trop importants pourraient décourager l'utilisateur, l'absence d'alternative utilisable pour explorer ces données les constraint toutefois à faire usage de la plate-forme d'exploration pour comprendre le comportement du modèle SimFeodal.

**Conserver la concentration** Le problème de la concentration de l'utilisateur demeure, lui, critique : des études ont montré, depuis longtemps (MACKENZIE et WARE 1993), qu'il y avait un lien fort entre la performance d'une interrogation visuelle et le délai nécessaire à son obtention. LIU et HEER 2014, p. 8 montrent ainsi qu'avec un simple délai de 500ms, la qualité des observations, des généralisations qui peuvent être tirées des données, et des hypothèses émises, décroît nettement. Les auteurs indiquent d'ailleurs que ces effets sont plus importants encore quand l'exploration est effectuée par des actions de *brushing* et de sélections croisées (*linking*), deux méthodes qui sont au cœur de SimEDB : « For example, more aggressive caching or prefetching methods may be employed for operations sensitive to small variations in latency, such as brushing and linking »(LIU et HEER 2014, p. 9).

FORCH et al. 2017, pour leur part, étudient la perception du délai de réponse lors d'interactions menées avec une souris d'ordinateur. Ils concluent ainsi que les utilisateurs perçoivent des délais d'attente inférieurs à 100ms, mais notent que les utilisateurs n'en sont pas pour autant perturbés, en particulier ceux qui ont le moins l'habitude de réactions rapides<sup>30</sup>.

Concernant le champ, plus spécifique, des *visual analytics*, nous n'avons pas trouvé d'articles de référence permettant d'établir une comparaison de l'efficacité des résultats trouvés selon la latence de la réponse. Les auteurs de ce champ recommandent de prêter attention à la rapidité de rendu et à son optimisation, mais sans que ne trouvions de résultats plus précis :

« When simple pattern finding is needed, the importance of having a fast, highly interactive interface cannot be emphasized enough. If a navigation technique is slow, then the cognitive costs can be much greater than just the amount of time lost, because an entire train of thought can become disrupted by the loss of the contents of both visual and nonvisual working memories. »

WARE 2012, tiré de AMIRPOUR AMRAII 2018, p. 12.

Tout au plus pouvons-nous émettre l'idée qu'il serait évident que la latence acceptée dans un environnement graphique de ce type soit largement supérieure à celle des environnements virtuels (réalité augmentée, visualisations immersives...), sans pour autant que nous ne puissions quantifier cet écart. SHNEIDERMAN et PLAISANT 2004, p. 74 indiquent tout de même, parmi les « 8 règles d'or du design d'interfaces », un délai maximal pour une phase d'exploration, sur lequel

30. Ils remarquent ainsi que les utilisateurs plus habitués à des jeux vidéos rapides (« highly dynamic computer games, such as action games, racing games, or first person shooter games [...] », FORCH et al. 2017, p. 51) sont plus vite affectés par le délai de réponse que les autres.

nous pouvons construire une estimation : « *Reduce short-term memory load*. The limitation of human information processing in short-term memory (**the rule of thumb is that humans can remember “seven plus or minus two chunks” of information**) requires that displays be kept simple, multiple-page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions. ». Si l'on considère qu'une session d'exploration requiert l'étude d'une dizaine d'indicateurs, et que chacun demande une quinzaine de secondes d'analyse visuelle, alors le temps d'attente entre les visualisations ne doit pas dépasser deux à trois secondes.

Selon ces différentes considérations, dans le cadre de SimEDB, on doit donc viser à développer une plate-forme aussi rapide que possible, tout en sachant, dès le départ, qu'il sera impossible d'arriver aux délais de 100ms ou 500ms évoqués précédemment, ne serait-ce que parce que le temps de requête des données – sans compter le temps de rendu graphique – est déjà supérieur d'un ordre de grandeur.

#### 5.4.1.3 Interopérabilité et évolutivité

Une autre contrainte forte tient cette fois au choix de l'environnement informatique qui accueillera la plate-forme d'exploration. On peut résumer ce choix à deux alternatives : un environnement local, en installant l'application sur l'ordinateur de chaque utilisateur, ou un environnement distant, où l'application serait donc accessible à distance, par exemple via une interface web.

Ce choix a des nombreuses répercussions, aussi bien en matière de possibilité d'accès que de facilité à faire évoluer la plate-forme. Le choix le plus classique est de développer une application installable sur un ordinateur : cela permet de garantir une utilisation à tout moment, sans contrainte d'accès au réseau internet, et cela permet aussi d'obtenir de meilleures performances, puisque la rapidité de l'application dépendrait uniquement de la puissance de l'ordinateur plutôt que de devoir souffrir du passage par l'intermédiaire d'un serveur.

*introduire le Mais déjà ici !*

**Différents supports d'interrogation** La performance d'une application locale, par rapport à une application distante, est un atout extrêmement intéressant, comme on vient de le montrer plus haut. Pourtant, cela implique une énorme contrainte : l'application doit être interopérable entre les différents systèmes d'exploitations (*Operating System*, OS) et versions de ceux-ci. Les utilisateurs potentiels de SimEDB, représentation fidèle des acteurs de la recherche, se partagent ainsi entre les trois systèmes d'exploitations majoritaires (Windows, MacOs, Linux).

Pour permettre à chacun d'utiliser SimEDB, il faudrait donc que le développement de cette plate-forme soit compatible avec ces différents OS, ce qui est une contrainte considérable en développement logiciel.

Ne mentionnons même pas les nouveaux OS, centrés autour d'usages tactiles, tels qu'on les retrouve sur les tablettes et autres *smartphones*, qui demandent, eux aussi, de nombreuses spécificités de développement.

En somme, disposer d'une application locale universelle, c'est-à-dire utilisable quelque soit le support informatique, est une quasi-impossibilité technique, et un objectif en soit, que notre travail de recherche ne cherche aucunement à résoudre. Pour garantir la faisabilité d'une plate-forme d'exploration de données locale dédiée aux données de simulation de SimFeodal, il faudrait donc commencer par restreindre son champ d'application à un ou deux supports officiels, par exemple l'OS Windows, abandonnant de fait les utilisateurs potentiels ne disposant pas de cette architecture logicielle.

) estime comment  
fonctionne ?

) déjà évoqué.  
+ haut  
(S.O.?)

avec quel  
inventaire  
on était pour  
BD particulièrement  
si oui, peut-être  
le dire à l'instant  
OK

OK

**Gérer les mises à jours et modifications** Comme pour les bases de données (Stockage distribué ou centralisé, page 27), la question de l'application locale ou distante pose un contrainte supplémentaire en matière de maintenabilité et d'évolutivité de la plate-forme choisie : dans le cadre d'une application locale (correspondant au distribué en SGBD), la distribution des différentes mises à jour de l'application entraînent nécessairement l'installation locale, à chaque fois. Le risque est alors que tous les utilisateurs ne disposent pas d'une même version, ce qui peut entraîner, par exemple, des contradictions dans l'évaluation d'expériences, certains utilisateurs ayant accès à une version « buggée ».

Sans aller jusqu'à ces extrêmes, notons qu'avec une application locale, le temps de répercussion d'une modification du code de la plate-forme est plus important : il faut en effet réinstaller sur chaque poste le logiciel ainsi modifié. Cela disqualifie de fait des modifications « en direct », par exemple lors d'une session collective d'exploration des résultats où les utilisateurs auraient des propositions de modifications à faire, ne serait-ce que pour des changements aussi infimes que des titres de graphiques ou d'axes.

**Le choix d'une application web** Au contraire, avec une application distante, donc basée sur l'accès, par un navigateur internet, à une application centralisée, ces problèmes ne se posent pas : des navigateurs sont disponibles pour tous les OS existants (OS dédiés aux ordinateurs ou aux usages mobiles), et interprètent de la même manière une page web, indépendamment de leur support de consultation. De plus, comme pour les SGBD, l'usage d'une plate-forme distante permet une répercussion instantanée des mises à jour et corrections : un utilisateur n'a qu'à rafraîchir sa page pour que la dernière version de l'application s'affiche. De la même manière, si un utilisateur souhaite étudier un nouvel indicateur, non prévu auparavant, le temps de déploiement peut être suffisamment court pour que cela soit possible au cours d'une même session d'exploration de données.

Il y a toutefois un désavantage certain, puisque les données permettant l'affichage des indicateurs doivent transiter sur le réseau internet : l'application n'est donc pas disponible hors-connexion, et, en cas de connexion lente, sera particulièrement éprouvée. Cette lenteur relative est toutefois compensée par un avantage à la centralisation de l'application : les calculs, parfois lourds, ne reposent pas sur les capacités individuelles des ordinateurs clients. En installant l'application sur un serveur dédié, il suffit donc d'augmenter les caractéristiques de celui-ci pour que les performances soient améliorées pour chacun des utilisateurs de l'application.

Dans le cas de SimEDB, nous disposons de ressources informatiques largement suffisantes (serveur de calcul interne à l'UMR Géographie-cités dans un premier temps, puis serveur de calcul partagé via la « Très Grande Infrastructure de Recherche » – TGIR – Huma-Num) pour assurer une rapidité de traitement des données et ainsi permettre à l'application SimEDB de se dégager de ce « goulot d'étranglement » technique qu'aurait sinon éprouvée la plate-forme.

#### 5.4.1.4 Généricité de l'interrogation et indépendance aux données

La dernière contrainte, plus technique, tient au besoin de généricité d'une plate-forme d'exploration de données vis-à-vis des données qu'elle interroge. On a résumé les possibilités et choix effectués en matière de SGBD (section 5.3.1 : Assurer la capacité d'interrogation des données), et décidé de ne retenir que des SGBD permettant une interrogation standardisée via des connecteurs génériques et un langage universel (le SQL).

oui, //  
while !  
Avait pr'  
éti + ha ?

L'infrastructure de stockage et d'organisation des données a ainsi été conçue pour être aussi générique que possible. Encore faut-il que la plate-forme d'exploration de données soit elle aussi aussi générique que possible, et donc en mesure de profiter de l'universalité du SGBD choisi.

**Indépendance au support de données** Une contrainte forte est donc constituée par la capacité de la plate-forme à être indépendante de la source des données : quelque soit le SGBD choisi, les requêtes émises par la plate-forme doivent être les mêmes, sans requérir d'adaptations spécifiques en dehors de la désignation du lieu de stockage des données et des pilotes du SGBD.

Dans les faits, lors de la construction de SimEDB (cf. section 5.2 : Comment explorer les données de SimFeodal ?), plusieurs solutions de stockage de données ont été utilisées successivement : de simples fichiers csv au départ jusqu'au SGBD ultra-performant MapD, en passant par des solutions plus classiques intermédiaires (SQLite et MonetDB notamment).

Il n'était donc aucunement question d'avoir à modifier le code source permettant de générer les indicateurs depuis les données, mais au contraire, de s'assurer d'utiliser des bibliothèques logicielles indépendantes des données, c'est-à-dire capables d'exécuter les mêmes chaînes de traitements quelque-soit la provenance des données.

**Indépendance aux requêtes et modularité de l'implémentation** Pour garantir cette générnicité, il est donc nécessaire de s'assurer que le mode de communication de la plate-forme vers les données soit bien basé sur un langage universel : le SQL. Il convient donc de choisir un ensemble de technologies permettant de générer des requêtes SQL, quand bien même l'expression de ces requêtes elles-mêmes serait conçue dans un autre langage. Faire appel à un langage intermédiaire, générant du SQL en sortie depuis une entrée sous forme d'un « *Domain Specific Language* » (DSL) permet ainsi de bénéficier d'une part de l'universalité du SQL, et d'autre part, d'une syntaxe plus expressive que celle du SQL. Pour les requêtes complexes, le SQL tend ainsi à être peu lisible, les opérations s'emboîtant les unes dans les autres de manière très linéaires, et donc, souvent verbeuses. En SQL pur, il est donc peu évident de créer une implémentation modulaire d'une requête, c'est-à-dire permettant une factorisation des commandes et un paramétrage des entrées.

Les indicateurs de sortie de SimFeodal sont pourtant, on l'a vu, assez fréquemment basés sur le même type d'opération : variation du nombre moyen d'agents au cours du temps de la simulation, courbes rang-taille des hiérarchies d'un type d'agent etc. Dans le cas du premier exemple, en SQL, pour passer d'une requête permettant de récupérer le nombre de foyers paysans au cours du temps, groupés par année et avec un filtre sur certaines simulations, il ne faut que quelques lignes de code. Pour adapter cette requête à l'interrogation du nombre d'agrégats, dans les mêmes conditions d'agrégation et de filtrage, l'effort est minime, mais peu évident à paramétrier pour rendre cette démarche plus générique.

En utilisant un DSL, plus adapté à la manipulation de données qu'à la sélection de sous-ensembles, on gagne donc en modularité d'implémentation, et donc en ré-utilisation de fonctions plus génériques, ce qui permet de disposer d'un code-source plus robuste, ré-utilisable et évolutif.

## Conclusion : Vers une plate-forme web générique et intuitive

Il ressort de ces différentes contraintes et choix d'utilisation que les choix techniques, propres aux modalités d'interrogation des bases de données, mais aussi méthodologiques, au regard de la conception d'une interface graphique intuitive, imposent des restrictions assez conséquentes sur les possibilités techniques disponibles pour la conception de la plate-forme SimEDB. En premier lieu, on fait le choix de se tourner vers une plate-forme implantée sous forme d'application web, utilisable depuis un simple navigateur – donc inter-opérable entre les différents supports technologiques –, ce qui exclue de fait quantités d'outils, de logiciels et de bibliothèques logicielles pensées pour l'exploration interactive de données. On souhaite de plus que la plate-forme utilisée dispose d'une interface aussi épurée que possible, donc nécessairement très adaptée au cas particulier des données issues de SimFeodal, ce qui là aussi élimine un ensemble de solutions « clefs-en-main », par exemple conçues autour des « webSIG » ou de bibliothèques logicielles de visualisations interactives intégrées. L'utilisation de la plate-forme doit être aussi efficace que possible, en cherchant à minimiser les temps de latence entre sélection interactive et affichage des indicateurs en résultant. On devra donc privilégier des ensembles technologiques récents et performants, intrinsèquement dédiés à l'interactivité, au détriment de frameworks plus génériques, conçus pour une forte diversité d'usage plutôt que pour la tâche très spécifique que constitue la manipulation interactive de données. Enfin, il faut que cette solution, dans la mesure du possible, soit en mesure de proposer une syntaxe d'interrogation de données modulaire, factorisée, et plus expressive que le SQL qu'elle doit pourtant réussir à générer.

### 5.4.2 Construire une plate-forme interactive pour l'évaluation de SimFeodal

Dans cette dernière sous-partie, nous allons donc présenter les choix – techniques, esthétiques et interactifs – qui ont été adoptés dans la conception et l'implémentation de SimEDB. Nous les présentons ici de manière linéaire, dans l'ordre quasi-chronologique du développement, mais il est important de garder en considération que ces éléments sont intimement intriqués : un choix technique conditionne les types d'interactions possibles, l'utilisation de telle méthode d'interaction peut restreindre l'étendue des possibles techniques etc.

Notons enfin que l'application SimEDB présentée ici, aussi bien dans son usage que dans sa conception, représente un instantané de développement, qui correspond à la période de rédaction du présent chapitre : à l'instar d'un modèle, une plate-forme peut et doit évoluer pour s'adapter aux besoins de ses utilisateurs tant qu'elle est utilisée. Les technologies et choix esthétiques introduits n'ont pas toujours été présents, et auront sans doute à évoluer dans la suite de la « durée de vie » de SimEDB.

#### 5.4.2.1 Choix des technologies

Nous présentons ici les technologies mobilisées dans le cadre du développement de SimEDB. Le but n'est pas d'entrer dans les détails de l'implémentation<sup>31</sup>, mais bien de justifier et présenter les choix relatifs aux technologies employées,

<sup>31</sup>. Le code source de SimEDB – et l'historique de son versionnement – sont, pour cela, disponible en ligne sous licence libre, sur la plate-forme Github : [github.com/RCura/SimEDB](https://github.com/RCura/SimEDB)

TB main  
jour en 70  
vain phras,  
pour solvante  
fort utile  
efficac-  
apprendre et  
expéri que fin  
as finement  
expliqué !