Analyse des outils d'automatisation de Salesforce

Grâce à la création d'une application de gestion de contrats d'agence immobilière

THÈSE DE BACHELOR

FRANCIS RUCKSTUHL

Mai 2020

Superviseurs de thèse :

Prof. Dr. Jacques PASQUIER–ROCHA
Software Engineering Group

Pasqual Gremaud
Software Engineering Group

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Software Engineering Group
Department of Informatics
University of Fribourg
(Switzerland)

Table des matières

Liste des figures

1 Introduction

Ce travail de Bachelor vise à explorer les divers outils proposés par Salesforce, une plateforme américaine spécialisée dans les logiciels de relations client, pour développer une application web hébergée sur le Cloud. Pour se faire, un prototype de gestionnaire des contrats d'agences immobilières y a été développé de toutes pièces. Il s'agissait au départ d'un projet de start-up dont le but était d'offrir aux PME immobilières des solutions autonomes moins coûteuses que les grands logiciels actuellement présents sur le marché. Cependant, pour des raisons expliquées plus en avant, cela s'est avéré sans réel intérêt en Suisse. Aussi, afin de ne pas mettre un terme à tout le projet, il a été reconverti et adapté pour satisfaire aux exigences d'un travail de Bachelor.

Le document est divisé en quatre parties principales.

La première recouvre les notions théoriques concernant les logiciels de gestion des ressources d'une entreprise, le Cloud et une description de Salesforce. Cela permet de prendre conscience des implications liées à ce style d'application, qui requiert des notions tant d'ERP que de CRM, tout en étant disponible partout dans le monde grâce au Cloud. Ces informations sont également nécessaires pour comprendre les atouts de Salesforce et justifier le choix d'y développer une application.

La seconde partie parle du concept de l'application web créée pour ce projet. On y aborde les raisons qui ont fait pencher la balance en faveur de Salesforce pour son développement, ainsi que les résultats des études de marché qui ont chamboulé les plans de la start-up. De plus, des notions théoriques concernant les contrats utilisés par les agences immobilières y sont également décrites de manière succincte, avec pour objectif de déterminer les informations légales qui doivent figurer dans les contrats de l'application.

La troisième partie traite de Salesforce et de ses instruments. On y présente, d'une part, des ressources d'apprentissage mises en ligne par et pour la plateforme et, d'autre part, les outils intégrés qui permettent d'automatiser le système d'une web-app.

Enfin, la quatrième et dernière partie aborde en détail l'implémentation de l'application de gestion de contrat créée pour ce projet. Cela permet d'observer de manière concrète les outils, leur paramétrage, leur utilité et leurs limites. Cependant, le logiciel ainsi développé est loin d'être complet car il ne vise qu'à servir de support à la description des capacités de la plateforme. En l'état, il ne pourrait être concrètement utilisé par une agence immobilière.

2

Théorie générale

2.1 Enterprise Ressource Planning (ERP)

2.1.1 Qu'est-ce qu'un ERP ?

Depuis qu'elles existent, les entreprises n'ont d'autre objectif plus crucial que la gestion optimale de leurs ressources pour maximiser leurs performances tout en minimisant leurs erreurs et leurs coûts.

Ainsi, dans un premier temps, l'arrivée des machines à calculer dans les années 1940 ont permis à l'employé de bureau de décupler sa vitesse de traitement de l'information, montrant au monde que la coopération avec les machines dans le domaine du business allait être indispensable à l'avenir.

Dans un second temps, la démocratisation des ordinateurs dans les années 60 a permis l'utilisation de programmes intégrés pour gérer et contrôler les inventaires. Ce sont-là les premières traces des Manufacturing Ressource Planning (MRP), ancêtres des ERP. Ils sont alors très coûteux et seules les plus grandes entreprises peuvent se les procurer.

Au fil des années, de nouvelles fonctions sont venues s'attacher à ces programmes, comme la planification et la gestion de la production, de l'ingénierie, des finances ou des ressources humaines.

Vint ensuite l'âge de l'internet, où, pour la première fois en 1996, ce type de logiciel est proposé via la toile et non plus en installation locale sur les machines de l'entreprise. C'est à partir de ce moment-là que les MRP ont franchi le cap et sont devenus des Progiciels de Gestion Intégrés (PGI), ou Enterprise Ressource Planning (ERP) en anglais. On y ajoute d'autres fonctionnalités d'administration moderne, comme la gestion de la relation client, des chaînes d'approvisionnement ou de l'aide à la décision. On passe donc d'un système régissant certains domaines d'une entreprise à un système capable d'en gérer efficacement l'ensemble, de façon centralisée et connectée.

Enfin, depuis 2005 et jusqu'à maintenant, l'apparition de la technologie Cloud a grandement diminué les coûts liés à l'installation et à la maintenance d'un tel système informatique. Les petites et moyennes entreprises ont ainsi plus facilement accès à des logiciels permettant une gestion efficace de leurs ressources.

Ceci a pour effet d'augmenter considérablement la demande globale pour ces services, à un point que les marchés pour ces logiciels spécialisés deviennent extrêmement profitables.[1]

2.1.2 Architecture

Qui dit demande élevée sur le marché dit également grand nombre de clients différents dont il faut satisfaire les besoins. En effet, chacun a sa façon de travailler, de s'organiser, de gérer ses employés ou ses clients et d'atteindre ses objectifs financiers.

Il en résulte que chaque entreprise n'aura pas forcément besoin de toutes les fonctionnalités ajoutées au fil des ans aux différents ERP. Un cabinet de consulting en Ressources Humaines ne va pas s'encombrer du module de gestion d'une chaîne de production, par exemple.

C'est en se basant sur ce genre de postulat qu'une architecture particulière à ce type de logiciel a été élaborée, afin de permettre à tout un chacun de sélectionner les fonctionnalités dont il a besoin. Il en résulte un cercle vertueux : puisque désormais on ne paie que pour ce qu'on utilise, les coûts d'achat et d'utilisation de ces logiciels diminuent, ce qui les rend abordables à davantage d'entreprises et donc répartit leur coût de développement sur un plus grand nombre de clients, ce qui à son tour contribue à abaisser leur prix d'achat.

Une façon simple de comprendre cette architecture est de se l'imaginer à l'aide de Lego.

Une grande plaque vide représente la base du logiciel, sans fonctionnalités. Sur celle-ci, on peut ajouter à l'envi des « briques Lego » de forme et de taille différentes, appelées *modules*, pour obtenir une construction plus ou moins complexe et satisfaisant au mieux à nos besoins.

Chaque module couvre une fonctionnalité particulière, par exemple la gestion des finances ou des relations client. La « plaque centrale » assure la compatibilité des composants en fournissant une base de données commune. De ce fait, il est aisé de façonner un ERP en fonction des besoins particuliers de son entreprise simplement en ajoutant ou retirant des modules.

Pour résumer sommairement, un ERP est donc un logiciel créé autour d'une base de données unique sur laquelle on peut ajouter ou retirer à volonté des modules indépendants mais parfaitement compatibles. De plus, un moteur fonctionnel doit permettre aux processus des différents modules d'interagir entre eux sans problème.

[1] Des détails et des compléments d'information sur la naissance des logiciels ERP peuvent être retrouvés à l'adresse suivante : https://www.versaclouderp.com/blog/history-of-erp-systems/

2.1.3 Objectifs d'un ERP

L'un des buts que l'on souhaite atteindre en intégrant un ERP à son entreprise est de faciliter le travail de ses différents pôles tout en réduisant ses coûts. On compte sur l'utilisation de *workflows* pour assurer un meilleur suivi de l'information tout au long de son traitement dans les divers départements, ainsi que sur des processus de fond pour automatiser les tâches redondantes ou coûteuses en temps.

Ce genre de logiciel offre un autre avantage considérable : une base de données centralisée et uniformisée qui garantit l'interopérabilité des données. Celles-ci deviennent alors plus fiables et permettent une communication efficace entre les modules. Par exemple, cela permet au Département des Stocks de transférer sans erreur les détails d'une livraison simultanément au Département des Achats pour vérification de la commande et au Département des Finances pour le paiement de la facture.

Enfin, l'utilisation d'indicateurs clés de performance, ou *Key Performance Indicators (KPI) en anglais*, permet d'analyser facilement l'efficacité de son entreprise et de connaître ses points forts et ses points faibles en un coup d'œil.[2]

[2] Des compléments d'information sur les objectifs des ERP sont disponibles à l'adresse suivante : https://www.choisirmonerp.com/erp/definition-d-un-erp

2.1.4 Fonctionnalités

Les fonctionnalités d'un logiciel ERP sont nombreuses et touchent à tous les domaines des entreprises. Il est impossible de toutes les regrouper, mais une liste même non-exhaustive des principales options de gestion donne une vue d'ensemble des capacités de ces logiciels. En effet, il est notamment possible de gérer entre autres les domaines suivants :

- Comptabilité et Finance
- stocks
- Ressources Humaines
- relations avec les fournisseurs et approvisionnements
- vente et distribution de produits (en e-commerce ou en physique)
- gestion de projet.

Figure 1 : Les principaux modules d'ERP
Image reprise du site `https://existek.com/`

Dès lors qu'une entreprise atteint une certaine taille et qu'elle doit traiter de multiples domaines, elle se doit d'implémenter ce type de logiciel. Elle ne peut en effet tout simplement pas se passer d'un tel outil afin de contrôler la quantité phénoménale d'informations à gérer.[3]

[3] Pour une lecture plus détaillée concernant les modules d'ERP et leurs propriétés, se référer à l'excellent article https://existek.com/blog/erp-modules-main-features-functionality-and-workflows/

2.2 Customer Relationship Management (CRM) [4]

2.2.1 Qu'est-ce qu'un CRM ?

Les logiciels de gestion de relation-client, ou Customer Relationship Management (CRM) en anglais, ont d'abord été développés en tant que module de fonctionnalité pour les ERP. Leur but était, comme leur nom l'indique, de fournir aux entreprises des pistes d'accroissement des ventes en perfectionnant leurs relations commerciales.

Cependant, ce domaine n'a cessé de croître en se complexifiant, si bien que les limiter à de simples modules d'ERP ne suffit plus. Ils sont donc rapidement devenus un type de logiciel à part entière, totalement autonome. Du fait de leur origine, ils ont hérité d'une architecture système similaire à leur « parent », reposant également sur une base de données centralisée à laquelle s'ajoutent des fonctionnalités sous forme de modules.

Quatre types de logiciels de gestions peuvent alors être construits[5] :

1. les CRM purs, spécialisés uniquement dans la relation client
2. les CRM croisés ERP, dont la relation client est le domaine principal, mais qui prennent en charge certaines fonctions de la gestion d'entreprise
3. les ERP croisés CRM qui s'occupent surtout de la gestion d'entreprise, mais qui ont des capacités dans les relations client
4. les ERP purs, spécialisés uniquement dans la gestion de l'entreprise.

Les entreprises cherchant à obtenir les performances les plus élevées dans l'un ou l'autre des domaines de gestion choisiront le logiciel correspondant. Les autres, cependant, se tourneront vers les modèles hybrides, qui permettent de s'adapter et de se développer plus facilement. Certes, les fonctionnalités sont moins poussées ou maniables que celles des logiciels spécialisés. En revanche, il n'est pas nécessaire de changer entièrement de système pour s'orienter vers un nouveau domaine, il suffit simplement d'ajouter le module adéquat.

Le marché devenant de plus en plus friand de cette flexibilité, les fournisseurs spécialisés dans les logiciels d'un seul type se sont rapidement adaptés. C'est le cas par exemple de Salesforce, qui s'occupait au départ uniquement de CRM, mais qui a modifié ses outils de développement pour aussi permettre d'implémenter des fonctionnalités d'ERP. C'est grâce à cette adaptation qu'est né le projet pour ce travail de Bachelor.

[4] Les informations de ce chapitre sont directement tirées de l'excellent article suivant : https://www.appvizer.fr/magazine/relation-client/customer-relationship-management-crm/qu-est-ce-qu-un-crm#qu-est-ce-qu-un-crm-definition/

[5] Les différences et les points communs entre CRM et ERP peuvent être lus de manière plus détaillée à l'adresse suivante : https://www.sage.com/fr-fr/blog/choisir-logiciel-crm-erp/

2.2.2 Que réalise un CRM ?

Afin de remplir son objectif de maximisation des ventes grâce au perfectionnement des relations commerciales, un CRM doit être capable de prendre en charge un grand nombre de fonctionnalités. Dont notamment :

- l'automatisation des processus de vente, d'enregistrement de nouveaux clients dans la base de données et d'envoi planifié d'informations à ceux-ci pour maintenir une communication efficace
- le recueil, le tri et l'analyse à des fins de marketing ou de prévision des ventes, des informations des clients à partir de la base de données
- la facilitation du suivi des ventes pour le Département commercial.

Grâce à la récolte et à la centralisation des informations, une entreprise est capable de mieux connaître ses clients et de leur offrir une expérience personnalisée selon leurs préférences, dans ses magasins ou en ligne.

De plus, les outils d'analyse permettent un prospect efficace et la mise sur pied d'offres et de publicités durant certaines périodes favorables. Ils donnent également un moyen de fidéliser les clients existants en calculant les besoins actuels et les prévisions de la demande. En effet, s'il se sent écouté, un client aura tendance à le rester et à consommer. Il est bien connu qu'il est plus profitable de garder un client acquis que de vouloir en gagner un nouveau.

Les CRM sont donc des outils efficaces pour répondre aux besoins d'information des entreprises qui cherchent à satisfaire et conserver leurs clients, afin de rester compétitives. Cela est d'autant plus important que la course à la numérisation et à la performance devient omniprésente. Seuls ceux qui savent tirer pleinement profit de ces outils d'analyse réussissent à croître.

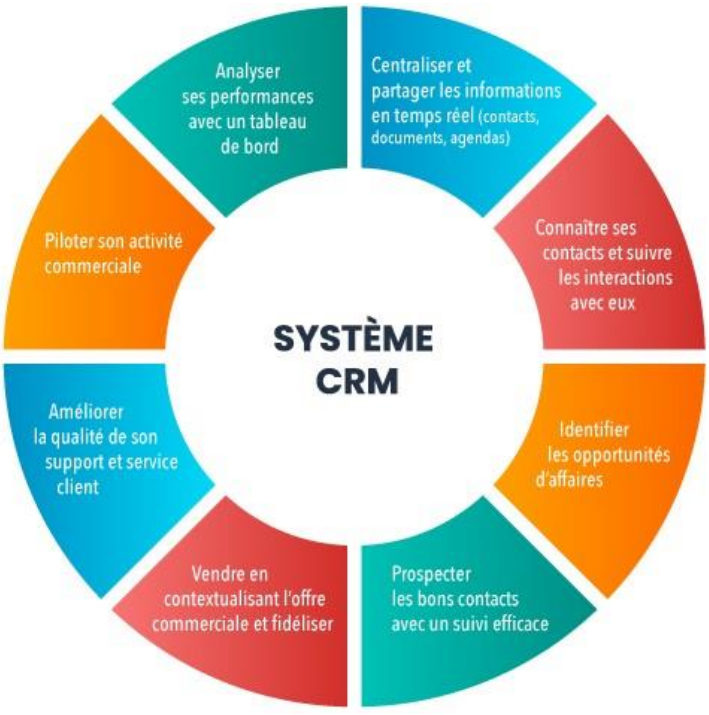


Figure 2 : Ce que permet de réaliser un CRM
Image reprise du site https://www.appvizer.fr/

2.2.3 Fonctionnalités

Pour être qualifié de CRM, un logiciel doit posséder certaines fonctionnalités indispensables. Elles sont au nombre de six :

1. la gestion des contacts à l'aide d'une base de données unique. Tout comme pour les ERP, c'est la base même du système, sans laquelle il n'a aucune raison d'exister
2. la centralisation des documents grâce à la technologie Cloud. Cela permet tant aux clients qu'aux employés d'interagir sur le même système en permanence et à travers le monde
3. la gestion des processus de vente qui procure un maximum d'assistance aux commerciaux dans leurs prospections
4. un outil de création et mise en place de campagnes de marketing ciblées.
5. l'analyse de performance au moyen d'indicateurs clés (KPI), pour renseigner en un clin d'œil sur l'activité commerciale de l'entreprise
6. la prise en compte du service client, notamment grâce à un support en ligne, pour satisfaire au mieux le client lorsqu'il a des questions ou un problème.

En fonction des besoins spécifiques de chaque entreprise, ces fonctionnalités indispensables peuvent bien entendu être complétées par des modules complémentaires.

2.3 Cloud

2.3.1 Qu'est-ce que le Cloud ?[6]

Dès l'apparition de l'internet, les ingénieurs informaticiens se sont rendu compte que ce moyen de transmission d'information pouvait grandement aider le monde du travail, notamment dans l'exploitation des données. Est né alors juste avant le XXI^{ème} siècle le *cloud computing*, un moyen de stocker des ressources et de proposer des services à la demande, grâce à internet.

On passe alors d'une technologie locale à une technologie à distance. Au lieu de garder ses données et ses programmes sur un disque dur de son ordinateur pour y accéder en permanence, il devient possible de recourir à une entreprise spécialisée dans le Cloud et de lui confier le tout. On garde bien entendu le contrôle sécurisé de toutes ses données, même à distance, et sans qu'il soit nécessaire de gérer une infrastructure locale.

De la sorte, entreprises ou particuliers peuvent louer des ressources de calcul et de stockage informatiques au prix d'un simple abonnement de service, lequel peut être ajusté à l'utilisation effective du Cloud. C'est là l'un des nombreux atouts de cette technologie. De ce fait, il devient bien plus profitable et pratique d'y recourir, plutôt que construire et maintenir sa propre infrastructure.

Pour preuve, le Cloud est devenu tellement omniprésent dans la vie de tous les jours que l'on ne se rend même plus compte de son utilisation quotidienne. Dropbox, Google Drive, la suite Adobe, Netflix, Skype, WhatsApp, Facebook ou encore Salesforce forment un modeste panel des programmes hébergés sur le Cloud auxquels on accède souvent quotidiennement.

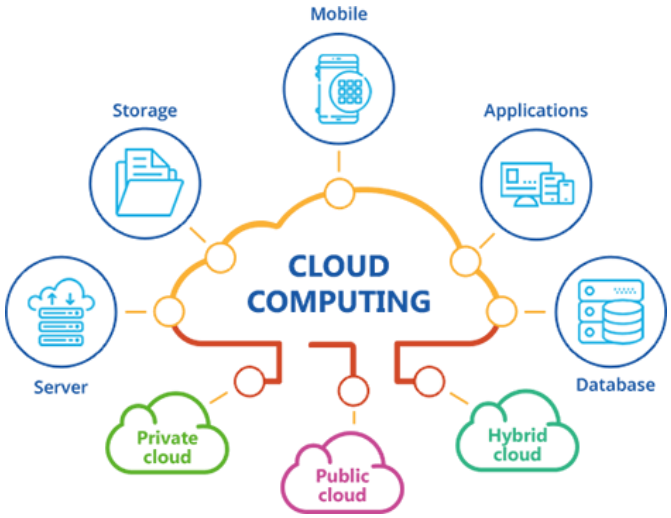


Figure 3 : Les vastes utilisations du Cloud
Image reprise de : https://www.macmillaneducationbookstore.com/

[6] Des informations plus détaillée sur le Cloud peuvent être obtenues en se référant à l'article suivant : https://www.culture-informatique.net/cest-quoi-le-cloud/

2.3.2 Structure

Le Cloud se décompose en cinq caractéristiques principales, quatre types de déploiements et trois modèles de services. Cela permet d'adapter son utilisation, que ce soit pour les loisirs ou le travail, en profitant des capacités de stockage ou de la puissance de calcul.

Les caractéristiques du Cloud :

Ce sont les fonctions minimales qu'un système doit être capable de fournir pour être considérée comme un *Cloud* :

1. un service à la demande, par exemple si l'on souhaite stocker des photos dans une banque de données en ligne
2. l'accès *via* internet, par exemple aux photos une fois qu'elles y ont été téléchargées
3. la mise en commun des ressources, ce qui permet par exemple à différents utilisateurs de stocker leurs photos dans une même banque de données, mais dans une zone qui leur est attribuée en propre
4. la flexibilité des ressources, afin qu'un client puisse augmenter en quelques clics son espace alloué pour y stocker ses toute nouvelles photos
5. un service quantifiable, qui permet à l'hébergeur du service de communiquer et facturer à ses clients la quantité d'espace disque qu'il leur loue.

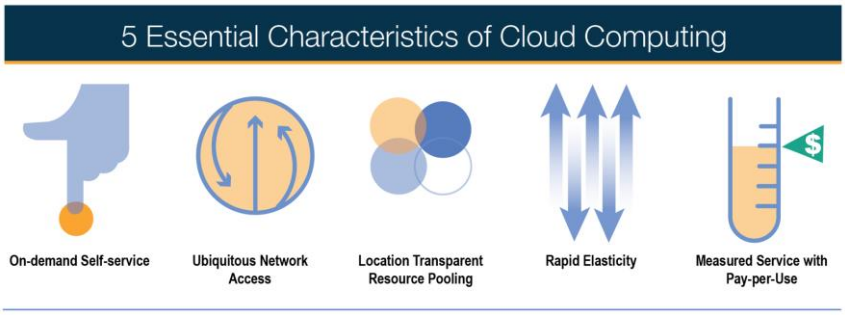


Figure 4 : Les cinq caractéristiques essentielles du Cloud
Image tirée de : https://packtpub.com/

Les types de déploiement :

Le Cloud peut être déployé et accessible de plusieurs façons, en fonction de son utilisation, regroupées sous ces quatre types principaux :

1. Cloud Privé : à disposition d'une entreprise ou organisation en particulier et accessible uniquement par elle
2. Cloud Public : n'importe qui peut avoir accès aux ressources, services et autres fonctionnalités
3. Cloud Hybride : un mix entre privé et public, pour permettre de manipuler des données sensibles dans un environnement contrôlé tout en utilisant la scalabilité du cloud public à bon escient.
4. Cloud Communautaire : le regroupement de plusieurs cloud privés.

Les types de services :

Il en existe différents types proposés *via* le Cloud, dont trois généraux qui représentent la majeure partie des possibilités. Des variations existent et des services très spécifiques peuvent être également offerts, mais ce sont-là des cas particuliers que l'on rencontre moins souvent :

1. « *Infrastructure as a service* (IaaS) », qui permet de posséder des serveurs dernière génération sans s'occuper des considérations matérielles, techniques et financières qui y sont liées. Il s'agit littéralement d'utiliser une infrastructure en tant que service
2. « *Platform as a Service* (PaaS) », qui offre la possibilité d'utiliser une plateforme semblable à une machine avec un OS. Ce service est généralement proposé par une entreprise mettant à la disposition de tout un chacun des outils tout prêts pour développer des applications sur le Cloud
3. « *Software as a Service* (SaaS) », un modèle récent de distribution de logiciel grâce auquel n'importe quel éditeur peut proposer à ses clients d'utiliser ses programmes, directement hébergés sur le Cloud.

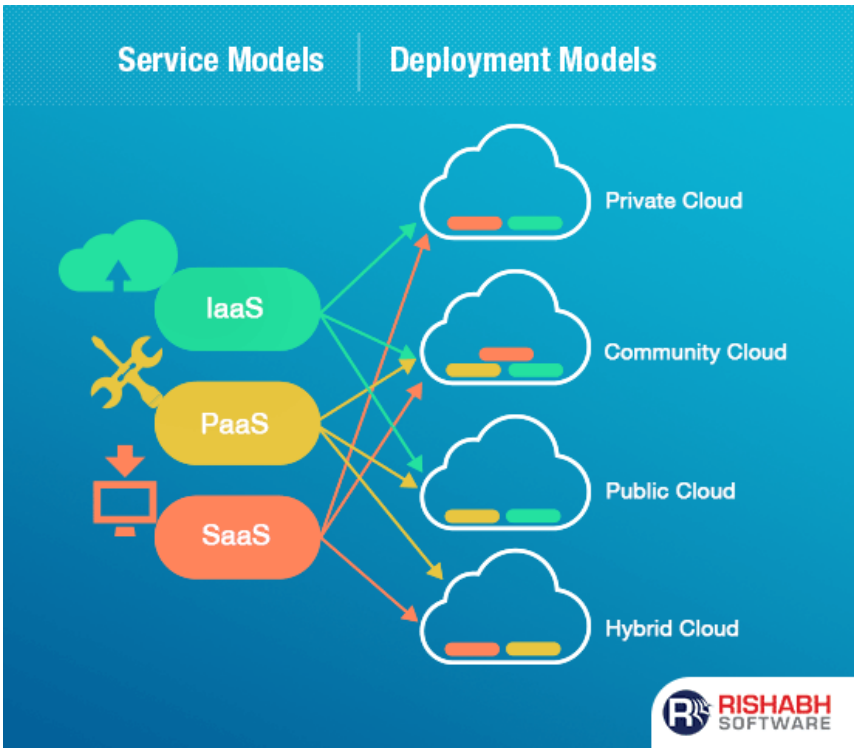


Figure 5 : Les trois types de services et quatre modèles de déploiement du Cloud
Image reprise de : https://www.rishabhsoft.com/

2.3.3 Avantages

En adoptant la technologie du Cloud, une entreprise ou un particulier accède à ces différents avantages :

- d'abord, puisque l'offre est accessible via internet, il n'y plus besoin d'une infrastructure locale et d'une équipe de support pour gérer tout le système. En effet, l'acquisition et la maintenance se font facilement en ligne, réduisant drastiquement les coûts informatiques en échange d'un abonnement de service
- ensuite, la connexion au logiciel peut s'effectuer à partir de n'importe quel endroit du globe, tant qu'il est relié à internet, et par des milliers, voire des millions d'utilisateurs en simultané
- enfin, les éditeurs ont élaboré des logiciels et solutions extrêmement flexibles en fonction des besoins de tout un chacun. Il est donc facile d'utiliser « à la demande » telle ou telle fonctionnalité à un moment précis pour ensuite s'en débarrasser sans aucun impact sur l'entièreté du système.

2.3.4 Inconvénients

Rien n'est parfait et la technologie Cloud n'échappe pas à cette règle. Elle ne présente que deux désavantages principaux, mais de taille :

- les clients sont entièrement dépendants des fournisseurs, tant pour la pérennité du service que pour la sécurité des données ou la mise en ligne de nouvelles fonctionnalités. Ainsi, une interruption du service ou d'internet peut se révéler catastrophique, surtout si elle dure plus de quelques minutes et qu'il est dès lors impossible de travailler pendant de longues périodes
- puisque les services ne sont accessibles qu'avec une connexion internet, il est nécessaire de posséder une bande passante suffisante, sans quoi la productivité s'en trouve impactée à la baisse[7].

[7] Des explications plus détaillées sur les avantages et les inconvénients du Cloud ainsi que sa structure générale peuvent être trouvées dans cet excellent article : https://www.lebigdata.fr/definition-cloud-computing/

2.4 Salesforce

2.4.1 Présentation

Salesforce est une entreprise américaine spécialisée dans la technologie Cloud, et plus précisément dans le domaine du PaaS et du SaaS. En effet, elle offre à ses clients des outils extrêmement poussés pour créer, puis héberger, une application web de CRM sans pour autant nécessiter d'eux des connaissances dans le domaine du codage. Le tout disponible sur sa plateforme en ligne accessible via le Cloud. C'est l'équivalent de WordPress pour les web-apps et la gestion des relations client.

Il s'agit là d'une formule qui fonctionne parfaitement, puisque l'entreprise est parvenue à devenir numéro un sur le marché international. En effet, ses outils se sont rapidement diversifiés pour donner la possibilité à tout un chacun, quel que soient les besoins et les contraintes, de développer l'application qu'il souhaite à travers Salesforce. En outre, une communauté mondiale s'est créée autour de la plateforme, permettant ainsi à de nombreux services complémentaires de voir le jour : support technique, forums, tutoriels[8], marché d'applications, emplois spécialisés dans ce type de développement, etc.

Il est à noter par ailleurs que le développement se fait directement en « responsive », permettant ainsi d'adapter automatiquement les applications aux tablettes et téléphones. De plus, la plateforme est maintenant tellement grande et diversifiée que certaines des web-apps qui y sont créées ressemblent fortement à des modules d'ERP, offrant ainsi des alternatives de gestion moins coûteuses aux petites et moyennes entreprises.

Figure 6 : Le logo Salesforce

[8] Tutoriels en ligne : https://trailhead.Salesforce.com/

2.4.2 Fonctionnalités :

Les fonctionnalités offertes par Salesforce sont bien sûr les six indispensables des CRM, à savoir la gestion de contacts, la centralisation des données, la gestion des processus de vente, des options de campagne marketing et des outils d'analyse de données et de maintien de relation-client.

Elles s'utilisent grâce à de nombreux outils de gestion de workflow, de processus, de base de données, d'hiérarchisation d'employés, de présentation visuelle, d'intelligence artificielle, d'analyse d'opportunités et de cycles de vente, et de bien d'autres. Le tout bien sûr de façon centralisée, pour permettre à un groupe de personnes de travailler en même temps et de manière efficace, tant pour utiliser que pour développer une application. C'est donc la plateforme idéale pour créer soi-même son logiciel CRM, ses petits processus automatisés ou gérer ses relations clients.

De plus, si l'on cherche à implémenter des fonctionnalités trop avancées pour être développées par ses propres moyens, le marché des applications[9] permet d'acquérir divers modules, de la même façon que l'on achèterait un jeu sur l'Apple Store ou le Play Store d'Android.

Figure 7 : Les domaines cibles de Salesforce
Image reprise de : https://www.talentpeaks.com/en/salesforce-platform/

[9] Marché d'application : https://appexchange.Salesforce.com

3

Concepts de la web app

3.1 Introduction

Ce projet de développement d'une application tire son origine de la volonté de découvrir le monde de l'entreprenariat. En effet, pourquoi ne pas commencer avec la création d'une solution informatique qui permettrait aux petites et moyennes entreprises immobilières de gérer leurs contrats plus facilement ? Ce domaine semble prometteur et la tâche est à la portée d'une ou deux personnes. Cependant, une étude de marché en profondeur a révélé certains paramètres qui n'avaient pas été pris en compte initialement et qui ont conduit à l'abandon de son développement à but commercial.

Aussi, afin de capitaliser sur les nombreuses heures déjà investies, le projet a été reconverti afin de démontrer une partie des capacités de Salesforce, la nouvelle plateforme de développement moderne sur le Cloud.

3.2 Concept

Le but de l'application est d'offrir à ses utilisateurs, en l'occurrence les parties prenantes des agences immobilières (CEO, managers, employés, clients, etc.), diverses possibilités de gestion et de stockage des contrats. En effet, un grand nombre de ces entreprises utilise des systèmes souvent incomplets, aux fonctionnalités démodées et généralement pour un coût exorbitant. Pire encore, certaines utilisent encore le papier pour la majorité de leurs démarches, avec tous les inconvénients écologiques, organisationnels et de stockage qui en découlent.

Ainsi, cette nouvelle web-app leur permettrait de créer en quelques clics un contrat immobilier, de le remplir avec les informations du client présentes dans la base de données, de surveiller la progression des négociations jusqu'à son approbation et sa signature, d'exporter si besoin au format PDF n'importe quel contrat ou encore de gérer l'ensemble des relations et des communications avec les clients. Le tout bien entendu accessible via le cloud depuis n'importe quel endroit et offert à des tarifs abordables pour les PME. Cela s'apparente donc à un module de système ERP de gestion de contrat sur une base CRM classique.

En effet, il s'agit de reprendre les deux fonctionnalités principales d'un ERP, la base de données centralisée et la possibilité de monitorer l'avancement d'un travail à l'aide de workflows et de processus, et d'ajouter à cela les fonctionnalités de relation-client des CRM. L'objectif est d'obtenir une solution logicielle flexible, fiable et facile d'utilisation pour tout un chacun.

3.3 Options de développement

Pour parvenir à créer une web-app avec autant de fonctionnalités différentes il est nécessaire de choisir un bon outil de développement. Voici les trois alternatives considérées pour ce projet :

1. Utiliser un Framework offrant des options avec le Cloud, comme par exemple *Ruby on Rails*. Cela permettrait un contrôle total sur l'application web en la développant à partir de zéro. Cependant, cela nécessiterait l'apprentissage dudit Framework, qui peut s'avérer long et complexe. Dès lors, cette tâche serait difficilement réalisable par une seule personne et ne convient donc pas forcément à ce genre de petit projet.
2. Créer un site internet à l'aide de WordPress ou tout autre CMS et l'adapter pour en faire une web-app. A l'inverse de l'option Framework, ce serait facilement réalisable par une personne seule, mais les fonctionnalités seraient limitées et forcément moins performantes ou flexibles que souhaité. De plus, les outils sont loin d'être adaptés et beaucoup de compromis devraient être acceptés.
3. Se tourner vers les options PaaS et SaaS de Salesforce, qui permettent de créer directement une application web sur le Cloud avec des outils préexistants. De plus, le déploiement sur la plateforme peut se réaliser directement après son développement sans devoir effectuer de migration.

Puisque Salesforce est pensé pour ce genre d'application et que le développement y est grandement facilité, la troisième alternative semblait la plus pratique et a donc été adoptée. De plus, ses outils peuvent être rapidement pris en main grâce à ses tutoriels en ligne, ce qui permet d'avoir un prototype concret dès le début du développement.

Seul inconvénient, la plateforme tourne principalement autour des fonctionnalités des CRM. Mais la flexibilité des outils permet de s'adapter et de contourner en grande partie ce désavantage. De plus, dans le cas où l'on aurait besoin d'une fonctionnalité très spécifique, il est toujours possible de la coder « sur mesure » avec le langage Apex utilisé par la plateforme.

D'autre part, le marché des applications de Salesforce permet une distribution facile et peu coûteuse une fois le produit terminé, si jamais il atteint le stade de la commercialisation.,

3.4 Études de marché

Démarches

Dans le cadre de ce projet, deux études de marché ont été réalisées.

La première a été effectuée auprès de l'agence immobilière *De Rham,* lorsqu'est apparue l'idée de créer une application de gestion de contrats immobiliers. Il s'agissait surtout d'obtenir leur avis d'experts, de se renseigner sur les spécificités requises d'une telle application et d'avoir un aperçu des outils actuellement sur le marché.

Des questions concernant l'ERP utilisé, son hébergement, ses fonctionnalités, les coûts, la taille de l'entreprise, etc… ont permis de déterminer l'option de développement optimale et les notions théoriques sur lesquelles il était nécessaire de se former pour implémenter une solution efficace et utile. C'est à partir de ce moment-là que la formation sur Salesforce a commencé.

Après traitement de ces points et confirmation de la faisabilité d'un tel projet par un seul développeur, la seconde étude de marché a été réalisée, auprès d'un panel de plusieurs agences pour déterminer si les choix effectués jusqu'alors étaient les bons. Des entreprises de toutes tailles et de toutes origines ont répondu, telles Remax, Rilsa ou encore Régimo.

Les questions posées portaient sur la satisfaction à l'égard de leur ERP, les fonctionnalités manquantes, la complexité du processus de création de contrats ainsi que les coûts engendrés. Le but était double : connaître les fonctionnalités sur lesquelles mettre l'accent, et développer une solution correspondant aux attentes de clients potentiels.

L'enquête a également porté sur la connaissance que ces agents immobiliers pouvaient avoir de Salesforce : s'ils en avaient entendu parler, ce qu'ils en pensaient et s'ils seraient potentiellement intéressés par une application développée sur cette plateforme. Elle a révélé que la plateforme était très peu connue des interviewés et les réponses n'ont donc pas été déterminantes.

Résultats de l'étude de marché

Ces deux études ont permis de faire ressortir diverses observations, dont voici les plus importantes.

Premièrement, les logiciels utilisés actuellement sur le marché sont issus d'une multitude de fournisseurs provenant du monde entier : SAP (Allemagne), GaraioRem (Suisso-Germanique), IList (USA-Canada), ID-Régie (Suisse), etc. Cela veut dire qu'il n'y a pas de réel leader mondial, mais que la concurrence est fortement présente partout dans le monde ; le client à l'embarras du choix et peut sélectionner le logiciel qui convient le mieux à ses besoins. De plus, il existe une tendance certaine à privilégier des systèmes gigantesques et très coûteux pour gérer toute l'entreprise plutôt que de créer de petits logiciels autonomes spécialisés dans un seul domaine.

Deuxièmement, la satisfaction tirée des logiciels ERP est loin d'être totale, avec une note de 7/10 en moyenne. Certains ne permettent pas de stocker les contrats au format digital, parfois le module de gestion de contrat n'est même pas intégré à l'ERP et tout se fait sur papier ; pour d'autres, l'interface très ancienne n'est pas intuitive. Au vu des prix déboursés pour l'utilisation de ces logiciels, on s'attendrait à une satisfaction bien supérieure. Il y a donc bien des pistes à explorer pour apporter des solutions à certains problèmes.

Finalement, les agences immobilières mettent un accent fort sur la confidentialité des données, généralement très sensibles, comme les informations bancaires, les contrats, les listes de noms et de relations, etc. De ce fait, la sécurité est l'une de leur priorité.

Cette dernière observation s'avère être le problème numéro 1 dans l'idée de start-up et la principale raison pour laquelle ce projet ne serait pas commercialisable. En effet, son développement sur Salesforce offre beaucoup d'avantages mais comporte un inconvénient majeur : cette plateforme est basée, développée, et stockée aux USA. Les lois sur la protection des données y sont nettement moins strictes qu'en Suisse, et aucune grande entreprise n'acceptera de faire des concessions sur la sécurité de ses informations. Pour ne rien arranger, il est impossible de travailler sur une version personnelle de la plateforme et de l'héberger ensuite sur ses propres serveurs. Tout est fait en ligne et donc potentiellement accessible aux américains.

C'est pourquoi il a été décidé de modifier l'application initiale pour qu'elle satisfasse les besoins de PME qui ne peuvent pas se permettre l'achat d'une licence d'ERP ou de gigantesques systèmes de gestion. Il s'agit de procurer aux entreprises travaillant encore principalement sur papier ou avec la suite Office une alternative peu coûteuse pour l'administration, le stockage de données et la gérance. C'est donc l'équivalent d'un module ERP/CRM à utiliser de façon autonome, qui certes assure une sécurité moins élevée des données, mais qui coûte aussi bien moins cher.

3.5 Théorie des contrats

Afin de bien saisir les particularités de l'application créée dans ce projet, il est nécessaire d'expliquer les principes sur lesquels se basent les contrats. En effet, ils constituent le centre névralgique des activités des agences immobilières, qui en gèrent de toutes sortes.

3.5.1 Qu'est-ce qu'un contrat ?

Un contrat, au sens juridique du terme, est une manifestation de volonté réciproque et concordante, dont l'effet est désiré par les parties (il en faut deux au minimum, par exemple un acheteur et un vendeur). Il est donc nécessaire d'en décrire en détail l'objet et le but, de manière contextualisée en fonction du type de contrat.

En effet, chaque type nécessite un certain nombre de clauses particulières afin de cerner l'ensemble de leurs spécificités. Ainsi, un contrat d'achat et un contrat de conciergerie comporteront des clauses semblables, telles que la description des parties prenantes, mais également des clauses tout à fait différentes, par exemple le prix d'achat pour l'un et la liste des activités pour l'autre.

Par ailleurs, dans certains cas, les contrats doivent être rédigés sous une forme précise et notariés, sans quoi ils ne sont pas juridiquement valables.

3.5.2 Différents types de contrats d'agence immobilière

Il existe un nombre phénoménal de contrats différents, et ce n'est pas le but de ce travail que de parler de chacun d'entre eux. Cependant, il est nécessaire d'aborder les points essentiels sur lesquels reposent les contrats utilisés par les agences immobilières, pour mieux déterminer les besoins d'implémentation dans l'application web du projet.

Location

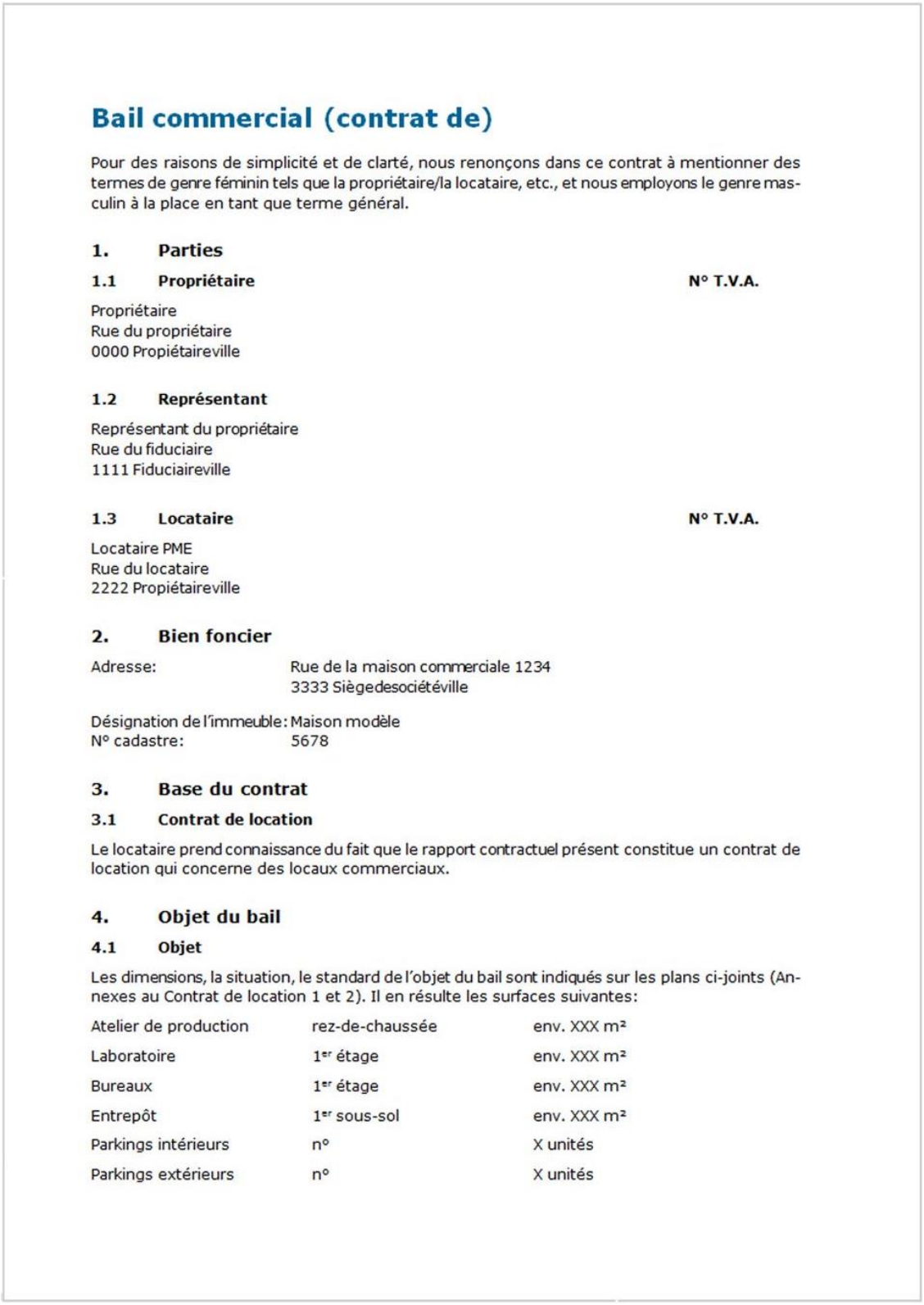
Le contrat de bail est un contrat par lequel une partie (le bailleur) cède à l'autre (le locataire) le droit d'habiter ou d'utiliser commercialement des locaux dont le bailleur est propriétaire. Il s'agira alors soit d'un bail d'habitation, soit d'un bail commercial. En échange, le locataire lui paye un loyer, dont le montant est déterminé dans le contrat.[10]

De ce fait, les clauses contenues dans un contrat de location sont au minimum :

- une description du bien loué
- des informations identifiant le bailleur
- des informations identifiant le locataire
- la durée du bail
- le montant du loyer

[10] Définition du contrat de bail, reprise des articles 253 et suivants du Code des obligations suisse

D'autres clauses exceptionnelles peuvent apparaître en fonction des particularités du bien loué, à son état, aux responsabilités des parties en cas de dégâts, etc. Ce sont des points généralement couverts de base par la loi et qui ne nécessitent pas d'être réécrits, hors cas particuliers.



Bail commercial (contrat de)

Pour des raisons de simplicité et de clarté, nous renonçons dans ce contrat à mentionner des termes de genre féminin tels que la propriétaire/la locataire, etc., et nous employons le genre masculin à la place en tant que terme général.

1. Parties

1.1 Propriétaire **N° T.V.A.**

Propriétaire
Rue du propriétaire
0000 Propiétaireville

1.2 Représentant

Représentant du propriétaire
Rue du fiduciaire
1111 Fiduciaireville

1.3 Locataire **N° T.V.A.**

Locataire PME
Rue du locataire
2222 Propiétaireville

2. Bien foncier

Adresse: Rue de la maison commerciale 1234
3333 Siègedesociétéville

Désignation de l'immeuble: Maison modèle
N° cadastre: 5678

3. Base du contrat

3.1 Contrat de location

Le locataire prend connaissance du fait que le rapport contractuel présent constitue un contrat de location qui concerne des locaux commerciaux.

4. Objet du bail

4.1 Objet

Les dimensions, la situation, le standard de l'objet du bail sont indiqués sur les plans ci-joints (Annexes au Contrat de location 1 et 2). Il en résulte les surfaces suivantes:

Atelier de production	rez-de-chaussée	env. XXX m²
Laboratoire	1er étage	env. XXX m²
Bureaux	1er étage	env. XXX m²
Entrepôt	1er sous-sol	env. XXX m²
Parkings intérieurs	n°	X unités
Parkings extérieurs	n°	X unités

Figure 8 : Un modèle de contrat de bail
Image reprise de : https://www.weka.ch/

Achat et vente d'un bien immobilier

Le contrat de vente immobilière est un contrat par lequel le propriétaire d'un bien immobilier (immeuble, appartement, maison, villa, ferme, entrepôt, etc.) cède son bien à un tiers (acheteur) contre payement d'un prix.[11]

Ainsi, ce type de document contient au minimum :

- une description du bien vendu
- des informations identifiant le vendeur
- des informations identifiant l'acheteur
- le prix de la transaction.

Il est à noter que ce type de contrat doit être conclu sous forme authentique, c'est-à-dire qu'il doit être rédigé par devant notaire (ou un autre officier public, en fonction du canton). Faute de quoi il n'est pas valable aux yeux de la loi. Ce point complexifie grandement le processus de création de façon informatisée. C'est pourquoi la gestion de ce type de contrat, même si elle fait partie des activités principales d'une agence immobilière, ne sera pas implémentée dans l'application.

Conciergerie

Contrairement aux contrats de location et d'achat/vente d'un bien immobilier, le contrat de conciergerie est « innommé ». C'est-à-dire qu'il n'apparaît pas expressément dans la loi et peut donc varier selon les volontés des parties. Néanmoins, lorsqu'il inclut un droit d'habitation, on le considère comme hybride entre un contrat de bail et un contrat de travail. [12]

En effet, dans ce cas il vise à rétribuer une personne pour son travail d'entretien d'un bien immobilier au moyen d'un droit d'habitation auquel s'ajoute, la plupart du temps, un salaire.

De ce fait, les clauses varient d'un contrat de conciergerie à l'autre puisque la loi n'en fixe pas expressément la forme. Il a été assumé pour ce projet qu'elles comportent au moins les éléments suivants :

- La description des locaux dont il faut prendre soin
- La description des tâches d'entretien à effectuer
- Des informations identifiant le propriétaire (ou le gérant) des locaux
- Des informations identifiant le concierge
- Une description des locaux mis à la disposition du concierge contre son travail
- S'il y a droit, le montant de son salaire monétaire.

Il convient de mentionner que de nos jours, il est devenu courant que le concierge soit un employé d'une entreprise spécialisée, liée au propriétaire de l'immeuble par un contrat de mandat. Il peut donc être intéressant de rajouter des informations à ce sujet dans l'application.

[11] Les informations sur les contrats de vente sont reprises du Code des obligations suisse, articles 184 et suivants.
[12] Se référer aux articles 319 et suivants du Code des obligations suisse pour plus de détails sur les contrats de travail

4

Développer sur Salesforce

4.1 Apprendre grâce à Trailhead

> *Trailhead est une plateforme d'apprentissage par l'expérience, une base de données de contenus éducatifs à laquelle vous pouvez accéder quand vous le souhaitez. Elle est conçue autour des besoins d'apprentissage des utilisateurs, et non des enseignements exigés par un service de formation*[13]

Trailhead est donc la plateforme d'apprentissage en ligne des outils de Salesforce. Elle permet d'approfondir ses connaissances du développement et d'en apprendre de nouvelles de façon amusante, didactique et modulaire. De fait, ses tutoriels se présentent sous forme de « jeux » indépendants les uns des autres qui rapportent des points. Ceux-ci sont ensuite utilisés dans un système de récompense de l'utilisateur pour le pousser à affiner ses connaissances dans tous les domaines, qui vont de la simple administration de la plateforme aux bases de l'intelligence artificielle, en passant par le cycle de vie d'une application. Pour se faire, la plateforme présente une structure particulière qui subdivise les catégories de compétences à assimiler.

4.2 Structure

Au sommet de la structure se trouvent les « Trails » ou parcours en français, qui représentent l'équivalent d'un gros chapitre, comme celui d'administrateur débutant [14](Figure 9). Il n'y a généralement aucune restriction concernant leur ordre d'apprentissage. L'utilisateur peut donc choisir n'importe quel sujet qui l'intéresse.

[13] Définition tirée directement de la page de support et de documentation de Salesforce pour *trailhead* : https://help.salesforce.com/articleView?id=mth_what_is_trailhead.htm&type=5

[14] Le trail et les modules qui sont abordés en tant qu'exemples peuvent être retrouvé à l'adresse suivante : https://trailhead.salesforce.com/fr/content/learn/trails/force_com_admin_beginner/

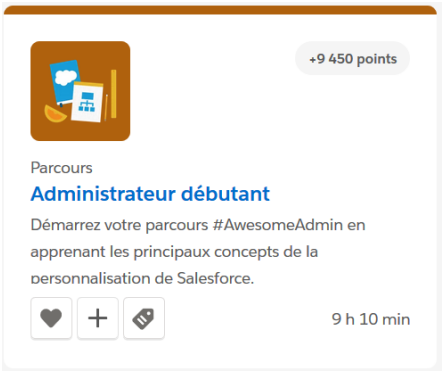


Figure 9 : Un exemple de Trail

Au sein de ces parcours on retrouve des modules, comme celui expliquant les concepts de base de la plateforme Salesforce (Figure 10). Une fois complété, il rapportera à l'utilisateur un badge en récompense de son apprentissage. Celui-ci est visible par toute la communauté et sert à témoigner de l'expérience d'un utilisateur vis-à-vis de la théorie.

Figure 10 : Un exemple de module d'apprentissage

Les modules sont ensuite subdivisés en unités d'apprentissage (à la Figure 11 il y en a 5) qui couvrent chacune une notion en particulier et se terminent par un défi pour s'assurer que les connaissances ont été acquises. Une fois toutes les unités terminées, le module sera complété et validé.

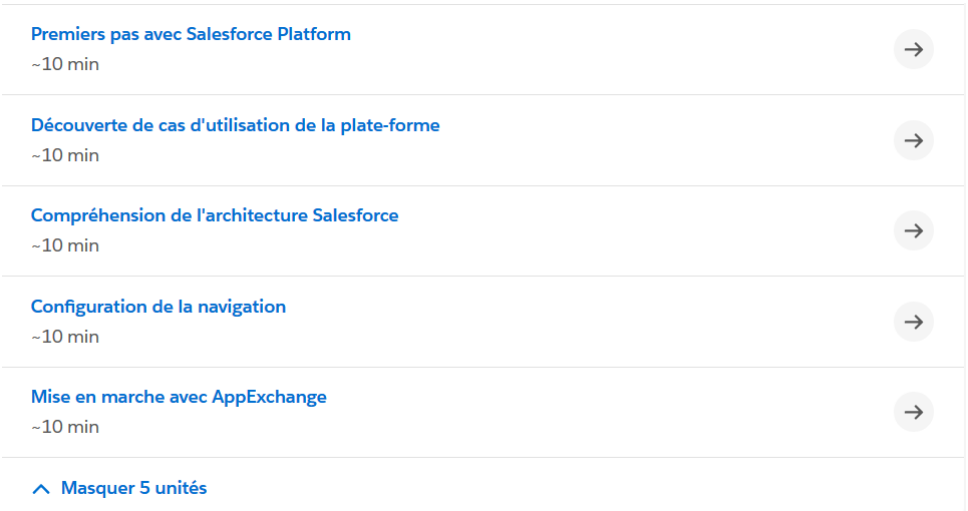


Figure 11 : Les 5 unités du module de la figure 10

Chaque défi des unités rapporte un certain nombre de points en fonction de leur nature : 500 points pour l'implémentation de fonctionnalité, comme c'est le cas à la Figure 12 ; de 200 à 50 points pour un QCM, en fonction du nombre d'essais pour le compléter. L'unité d'apprentissage est réussie et les points obtenus une fois le défi validé par le système.

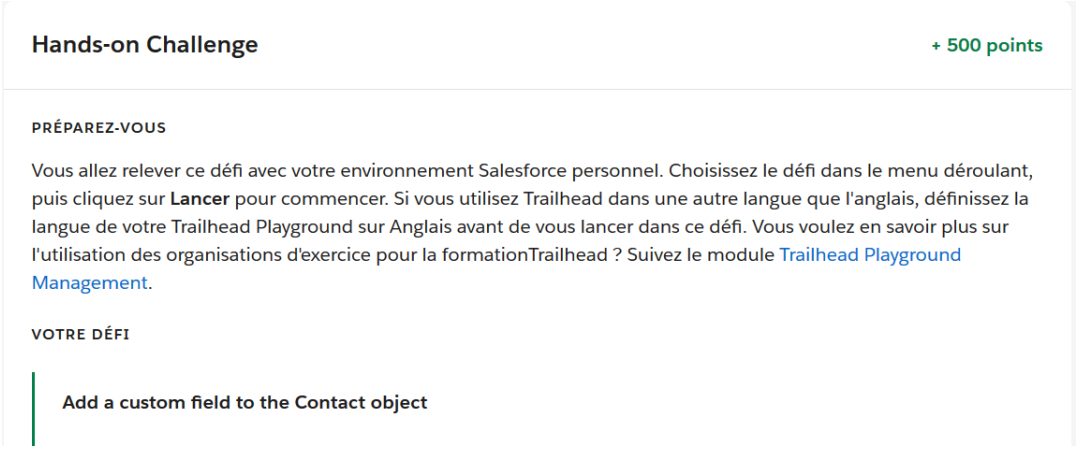


Figure 12 : Un exemple de défi

De plus, il existe un système de grade d'utilisateur en fonction du nombre de points et de badges qu'il a acquis, comme décrit à la Figure 13. Cela permet de montrer à la communauté l'expérience des utilisateurs avec la plateforme, tout en les motivant à faire de leur mieux pour se classer au plus haut.

Figure 13 : Un exemple de badge d'utilisateur

4.3 Les outils de Salesforce

Salesforce repose sur une terminologie un peu particulière. Il est donc nécessaire d'expliquer les termes utilisés et les divers outils de développement. Il est impossible de tous les aborder, au vu de leur nombre, sont donc décrits en détail uniquement les plus importants pour ce projet. Ces informations sont en partie reprises des différents « trails » proposés sur trailhead.com.

4.3.1 Interface utilisateur

Salesforce possède une interface utilisateur globalement identique pour chacune de ses applications. On y retrouve divers éléments facilement reconnaissables, comme on peut l'observer à la Figure 14. Ils sont décrits en même temps que la terminologie de la plateforme.

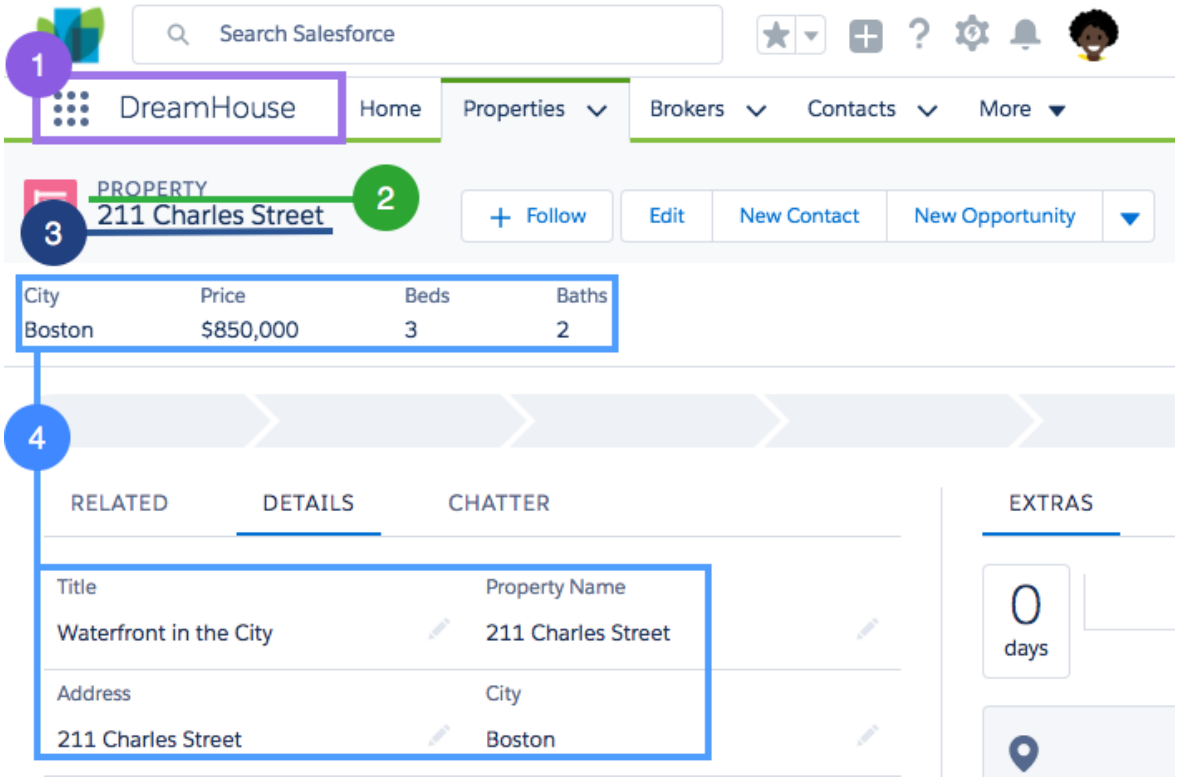


Figure 14 : L'interface utilisateur

1. Salesforce fonctionne sur un système d'applications. Chacune a ses propres fonctionnalités et agit sur ses propres objets. Il est possible d'en changer grâce à l'icône (⁞⁞). Cela permet, par exemple, de naviguer entre un module de gestion de contrat et un module de finance. L'application ici utilisée est nommée « DreamHouse ».

2. Les objets sont les tableaux dans la base de données Salesforce. Chaque objet stocke un seul type d'information en particulier et existe sous deux formes : standard et personnalisé. Dans le premier cas, les objets sont automatiquement disponibles lors de la création de l'application et se basent sur des fonctionnalités basiques (contacts, comptes, etc.). Dans le second cas, ils sont créés manuellement par l'utilisateur pour répondre à des besoins plus précis.
   Les pages d'information des objets sont représentées par les onglets où figure un menu déroulant (sorte de "V"). Chacune possède son propre *layout* et ses propres boutons. Attention cependant à ne pas les confondre avec les pages normales de l'application, comme la *home page* ou le *chat*, qui ne reposent sur aucun élément de la base de données. Néanmoins, elles n'ont pas de menu déroulant, ce qui permet de les distinguer facilement.
3. Les enregistrements ou entrées sont les lignes dans les tableaux de la base de données. Ce sont l'ensemble des détails associés à une instance d'un objet. À la Figure 15, la propriété 211 Charles Street est un enregistrement dans le tableau des propriétés, par exemple. Chaque enregistrement possède une clé primaire, qui sert à le distinguer.
4. Les champs sont des colonnes dans les tableaux de la base de données. Il s'agit des « détails » comme le prix ou la localisation d'un objet en particulier. Ils possèdent chacun un type, qui est définit lors de leur création et qui permet de restreindre les valeurs qui y sont inscrites. Ainsi, il n'est pas possible d'écrire du texte dans un champ définit pour des nombres.

De plus, des raccourcis qui facilitent la vie de l'utilisateur sont accessibles dans le header en haut à droite de l'interface. Ils permettent d'attribuer en « favoris » des onglets (étoile), d'ajouter rapidement des objets (petit "+"), de consulter ses notifications (cloche), accéder à l'interface de développement si l'on est administrateur (rouage) et consulter son profil (portrait).

4.3.2 Interface développeur

Lorsque l'on se forme sur *Trailhead*, une partie des défis demande d'implémenter des fonctionnalités en guise d'exercices. Pour se faire, un environnement de travail est donné à l'utilisateur lors de la création de son compte. Quoique fortement limité au niveau de la mémoire allouée, ce qui l'empêche d'être utilisé dans de grands projets, il reste tout de même en tout point semblable à celui fourni avec une licence Salesforce et suffit amplement pour se former sur la plateforme.

Comme on peut l'observer à la Figure 15**,** l'interface de développement consiste surtout en un menu (à gauche) à partir duquel on accède aux différents outils. Un champ de recherche est disponible pour faciliter la navigation.

La fenêtre centrale rassemble des postes d'aide et des tutoriels (présents uniquement dans l'environnement gratuit), ainsi qu'une liste des éléments récemment modifiés. C'est dans cette fenêtre qu'apparaissent les interfaces d'utilisation des outils une fois qu'on en sélectionne un.

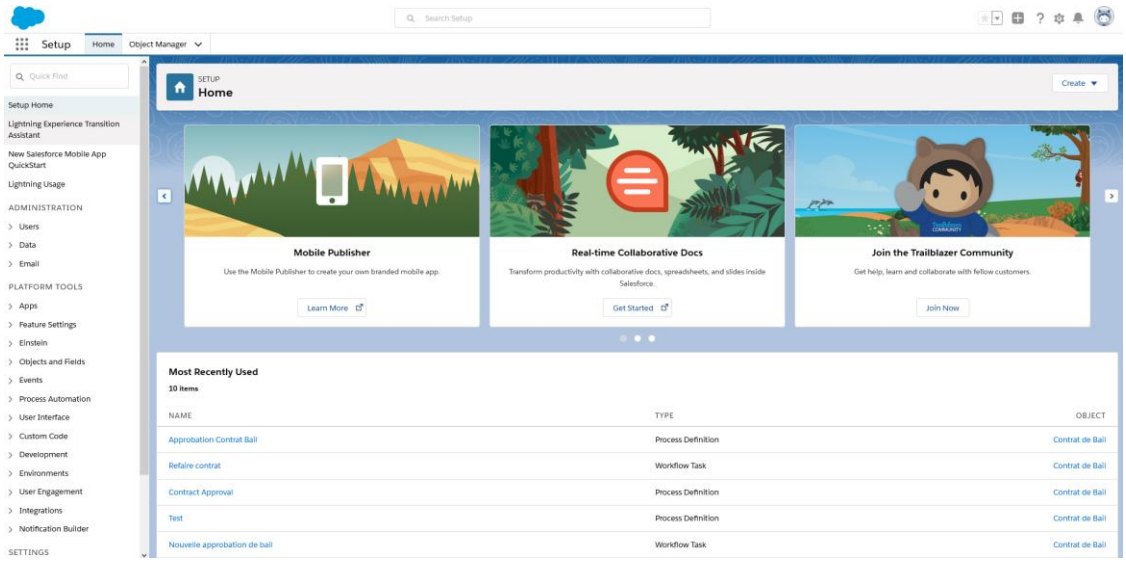


Figure 15 : L'interface d'administration

De façon non exhaustive et fortement résumée puisqu'il en existe énormément, les outils présents dans la liste de gauche permettent de gérer :

- les utilisateurs, leurs rôles, leurs permissions, etc.
- la taille, la mémoire, la sécurité, les données, etc. du système en général
- les Emails, avec la création de modèles, leur format, leur chiffrage, l'intégration Outlook, etc.
- les applications de l'organisation, le marché *AppExchange*, le format mobile, etc.
- l'outil *Einstein*, qui s'occupe de tout ce qui concerne l'intelligence artificielle, le *machine learning*, la prédiction d'opportunités, etc.
- les *Objets* de la base de données, leurs champs, leur disposition au sein d'une page, l'intégrité des données, etc.
- l'automatisation du système avec les *Flows*, les *Processus*, les demandes d'approbation, etc.
- l'interface utilisateur avec le *Page Builder*, la disposition des onglets, des menus, etc.
- l'ajout ou la modification de classes Apex, le langage informatique utilisé par la plateforme. Celui-ci ressemble fortement à du HTML / CSS orienté objet et permet de personnaliser le système et ses outils selon ses propres besoins, au lieu de travailler uniquement avec les modèles proposés de base.

La majorité des opérations de développement se font uniquement à la souris avec du drag & drop et au clavier pour remplir des formulaires. Il n'est donc pas nécessaire d'apprendre le langage Apex pour commencer à développer sur la plateforme, ce qui est l'un des principaux atouts de Salesforce. En effet, cela permet à tout le monde de créer une application et de l'automatiser sans devoir toucher à une seule ligne de code.

Tous les outils disponibles sont extrêmement intéressants et mériteraient chacun d'être abordés, mais ce sont surtout ceux d'automatisation qui vont être décrits, puisque ce travail se focalise sur eux.

4.3.3 Base de données

L'outil *Object Manager* fait partie des fondations de l'infrastructure Salesforce. Il permet aux développeurs de créer une base de données avec ses relations et d'assurer de façon automatique l'intégrité des données grâce à des formules ou à des règles de validation.

En terme simples, on pourrait comparer ces objets à des tables d'une base de données, puisqu'il existe une entité séparée pour chaque élément ou concept dont on souhaite stocker des informations. En réalité, c'est plus complexe que ça, puisqu'il est possible d'y créer de façon indépendante pour chaque objet une interface utilisateur, des processus, des workflows, des filtres de données, etc.

La création d'un nouveau champ d'un objet se fait à l'aide d'une liste exhaustive d'option pour définir son type (Figure 16). Il suffit ensuite de remplir des formulaires avec les détails de personnalisation, comme son nom, sa taille et les particularités du type (les multiples valeurs de la liste déroulante pour les *picklists*, le nombre de chiffres pour un champ numérique, etc.)

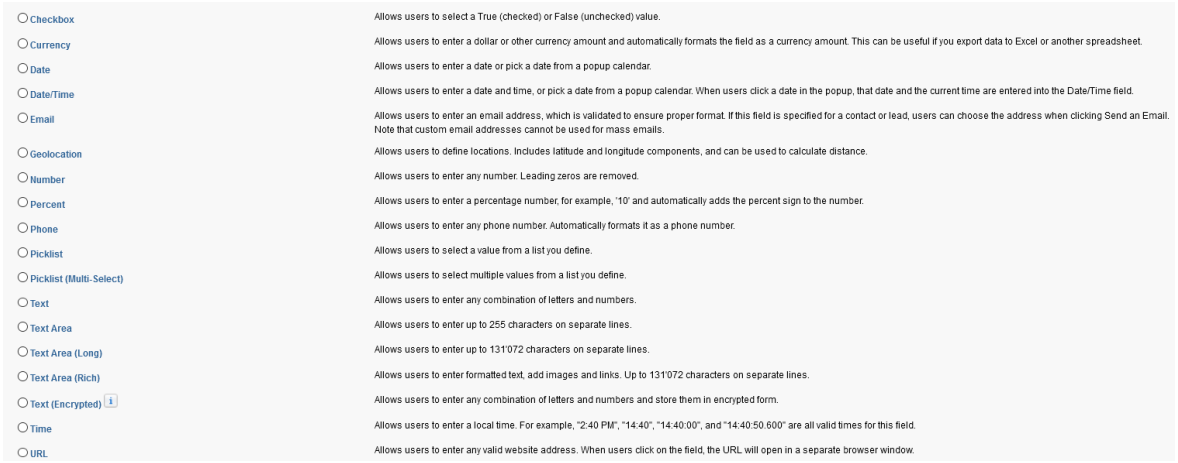


Figure 16 : Les différents types que peuvent prendre les champs

Les relations entre les objets sont réduites au minimum pour simplifier grandement le développement, mais sont construites de façon à pouvoir créer de gigantesques systèmes. En effet, il en existe deux types : les « look-up » et les « master-detail ».

La première relation permet à un objet de « regarder » dans un autre en se basant sur un système de clé étrangère. Elle peut être sous la forme [1 ; 1] et [1 ; multiple].

La seconde est une relation « parent – enfant » (ou maître – détail) entre deux objets. C'est-à-dire que l'un peut contrôler en partie le comportement de l'autre, par exemple ses accès à certaines données. Ce type de relation implique une dépendance totale des « enfants » en l'existence du « parent ». En effet, si ce dernier vient à être supprimé, ils le sont également.

Il est à noter qu'un « enfant » peut être lui-même le « parent » d'un autre objet, permettant l'existence de chaîne hiérarchique avec des « super parents ». De plus, un objet ne peut être qu'une seule fois enfant mais plusieurs fois parent. Ce type de relation peut donc s'avérer rapidement complexe à gérer à cause des différentes dépendances impliquées, mais permet néanmoins d'assurer une grande sécurité des données.

Pour ce qui est de la visualisation de la base de données, une interface (Figure 17) est intégrée directement à la plateforme et permet d'offrir au développeur une vue d'ensemble de tout le système. Les liens de couleur bleue représentent une relation de type "lookup" tandis que les rouges indiquent une relation de type "master-detail". De plus, un trait rouge dans la table signale l'obligation de renseigner ce champ si l'on souhaite créer une instance de l'objet.

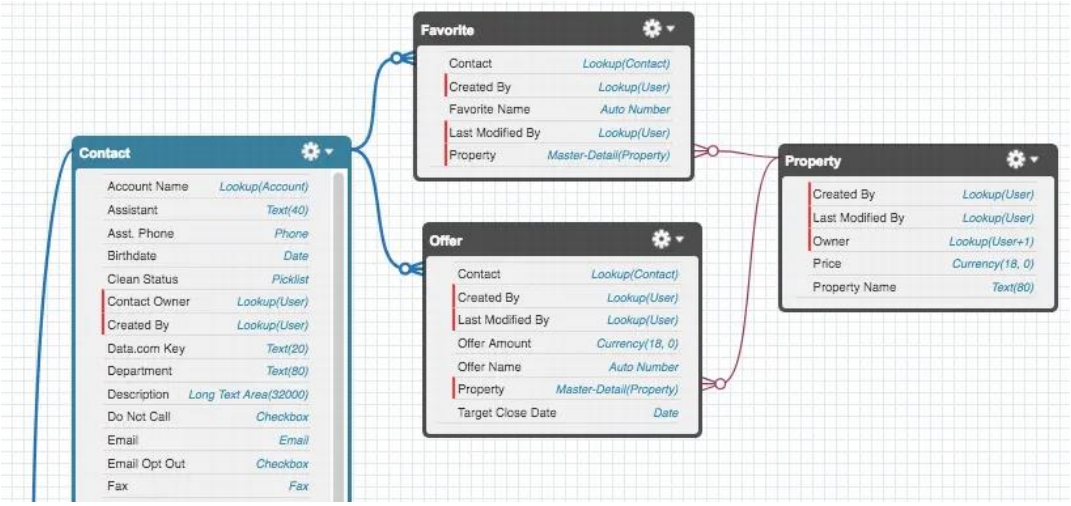


Figure 17 : La présentation visuelle d'une base de données

4.3.4 Automatisation

Salesforce possède plusieurs outils d'automatisation. La plateforme distingue ceux consacrés à l'expérience utilisateur (et donc apparaissant à l'écran) et ceux qui agissent en tâche de fond sur le système. De plus, il est intéressant de noter que du code Apex peut être appliqué à chaque outils si jamais ses capacités ne correspondent pas entièrement à nos besoins. C'est cependant une opération complexe, qui nécessite de très bien connaître la plateforme.

Constructeur de Workflow

Le premier outil disponible pour effectuer des tâches d'automatisation est le *Flow Builder* (constructeur de flux en français). Il permet de créer simplement des workflow (que Salesforce appelle plus simplement *Flow*) guidant l'utilisateur dans l'accomplissement de ses tâches.

Son objectif est de faire apparaître l'une après l'autre des informations, fenêtres pop-up, formulaires et autres données à l'écran de l'utilisateur tout au long de de son emploi de l'application. Par exemple, si la case [non] est cochée en réponse à la question « êtes-vous déjà client » d'une page de login, le workflow utilisera ce choix pour déclencher diverses actions, comme faire apparaître à l'écran une fenêtre d'inscription.

Le domaine d'influence des flows se limite quasi exclusivement à ce qui doit s'afficher à l'écran pour interagir avec un utilisateur, mais cela rend leur création extrêmement simple. L'utilisation de l'outil est donc très intuitive et offre la possibilité de mettre sur pied des petits ou des grands processus, quelles que soient ses compétences.

Le développement se fait par « drag & drop » à la souris uniquement. On place dans la fenêtre de développement les différents objets concernés, les actions à prendre, la logique à appliquer et les conditions à remplir pour avancer dans le processus. On y ajoute ensuite la notion de flux en les reliant par des flèches. Enfin, il suffit simplement d'activer le flux et de l'insérer dans une page à l'aide de l'outil *Page Builder* pour qu'il soit utilisable.

Il est à noter que les flows ne sont pas obligatoirement visibles. En effet, certains peuvent s'exécuter en arrière-plan sans interaction avec l'utilisateur. C'est le cas dans la Figure 18, tirée directement de Trailhead et où aucun élément "Ecran" ne fait partie du processus. Il est également possible de lancer des flux en parallèle ou en série grâce à l'élément "Subflow".

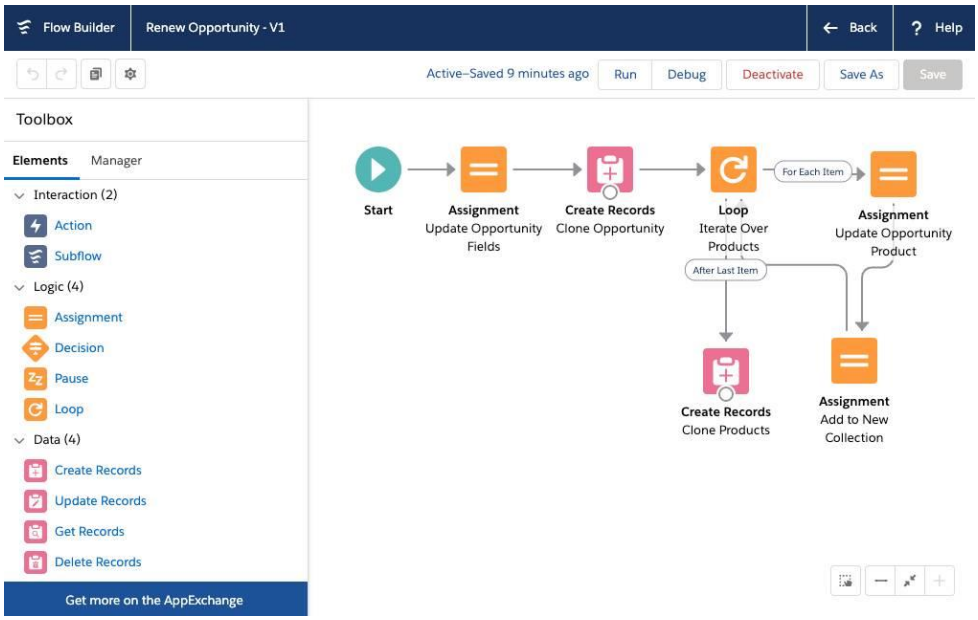


Figure 18 : Un processus du *Flow Builder*

Tous les éléments de la liste de gauche de la Figure 18 se configurent de la même façon, à quelques nuances près en fonction de leur rôle. Le plus intéressant est celui d'écran (n'apparaissant pas sur la figure). En effet, c'est grâce à celui-ci que l'on construit l'interface à travers laquelle l'utilisateur va interagir avec le système.

Pour ce faire, il suffit d'utiliser la souris pour glisser les éléments de la liste *d'input* de la Figure 19 dans le canevas au centre, qui représente ce que verra l'utilisateur. On peut observer dans cette même figure qu'un champ de type date y a été ajouté. On configure ce dernier dans la colonne de droite, où on lui donne premièrement un label et un nom d'API, ainsi qu'une valeur par défaut si elle est nécessaire. Ensuite, il est possible (mais pas indispensable) de déterminer les paramètres de visibilité du champ, d'y valider les inputs rentrés grâce à une formule de vérification, et d'ajouter un commentaire pour aiguiller l'utilisateur sur le format d'input à fournir.

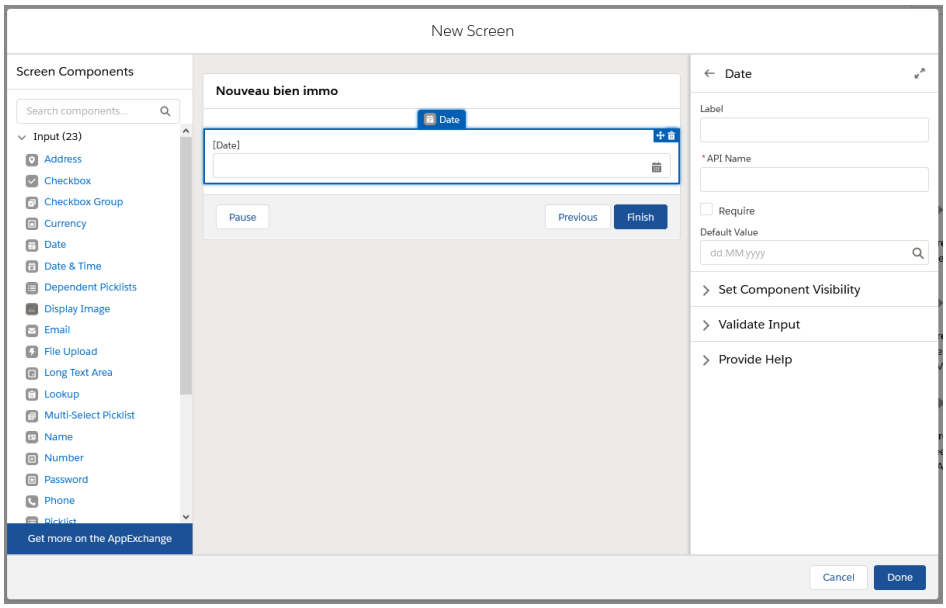


Figure 19 : Paramétrage d'un élément « écran »

Les autres éléments du flow sont configurés de la même façon, si ce n'est que les formulaires à remplir sont différents.

Par exemple, lorsque l'on utilise l'élément "Create Records" pour ajouter une entrée dans la base de données selon les inputs de l'utilisateur, il faut faire correspondre les noms d'API des champs de l'écran avec ceux de l'objet qu'on souhaite créer. Ainsi, à la Figure 20, le champ Adresse__c de l'objet *Bien Immobilier* va être renseigné par la valeur du champ Adresse d'un élément écran.

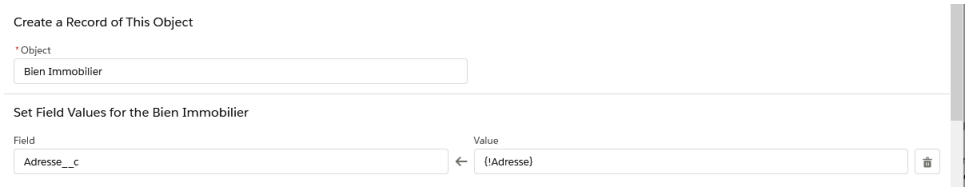


Figure 20 : Paramétrage d'un élément "Création d'enregistrement"

Constructeur de Processus

Les workflows sont utilisés pour interagir à l'écran avec l'utilisateur tandis que le *Process Builder* permet de créer des *Processus* pour effectuer des tâches en fonction de l'état du système. Ces processus ne peuvent être lancés que de trois façons différentes :

- lorsqu'un enregistrement est ajouté à la base de données ou modifié (on crée un nouveau contrat ou on en modifie un existant, par exemple)
- lorsqu'un événement particulier survient sur la plateforme (Une erreur du système ou d'un flow, par exemple)
- lorsqu'un autre *Processus* les invoque.

Cet outil est donc intimement lié aux objets de la base de données et au système en général, ce qui permet d'agir directement sur eux. Les actions réalisables, nombreuses et variées,

permettent d'interagir avec l'ensemble de la plateforme. De façon non-exhaustive, il est possible de :

- créer ou modifier une entrée dans la base de données
- ouvrir un script ou une classe Apex
- enclencher un *Flow* ou un autre *Processus*
- envoyer des alertes Email ou des notifications personnalisées
- ajouter un poste au forum intégré à l'application, le "Chatter"
- enclencher un processus d'approbation.

Par exemple, un client change son adresse dans l'objet « contact » puisqu'il vient de déménager. Au lieu de lui demander de la changer aussi dans l'objet « ville de livraison », comme cela se ferait avec un *Flow*, le système s'en occupe automatiquement grâce à un processus dédié enclenché dès lors qu'une modification est apportée au champ d'adresse de cet objet. Si le processus a été paramétré dans ce but, il envoie alors une notification à l'utilisateur pour lui signaler que les changements ont été correctement effectués.

De fait, on constate que les *Processus* créés grâce au *Process Builder* sont capables d'effectuer bien plus de tâches que ceux du *Flow Builder*. C'est pourquoi ce dernier n'est réellement utilisé que pour certaines tâches très précises ; la majeure partie de l'automatisation au sein des applications Salesforce se fait grâce au *Process Builder*.

Cependant, la prise en main de cet outil est plus complexe et l'implémentation de processus moins intuitive. Il est nécessaire de préparer d'avance certaines ressources, qui peuvent elles-mêmes demander une dose de planification, comme c'est le cas avec les requêtes d'approbation. Il est donc indispensable de bien connaître les capacités de la plateforme et le fonctionnement de ses autres outils.

L'interface de développement est relativement semblable aux *Flows*, du fait que tout se réalise à la souris et aucune ligne de code n'est requise. De simples clics permettent d'ajouter de nouvelles conditions et de déclarer les actions à effectuer selon qu'elles soient vérifiées ou non. Graphiquement, la notation est proche du BPMN mais reste tout de même bien moins poussée.

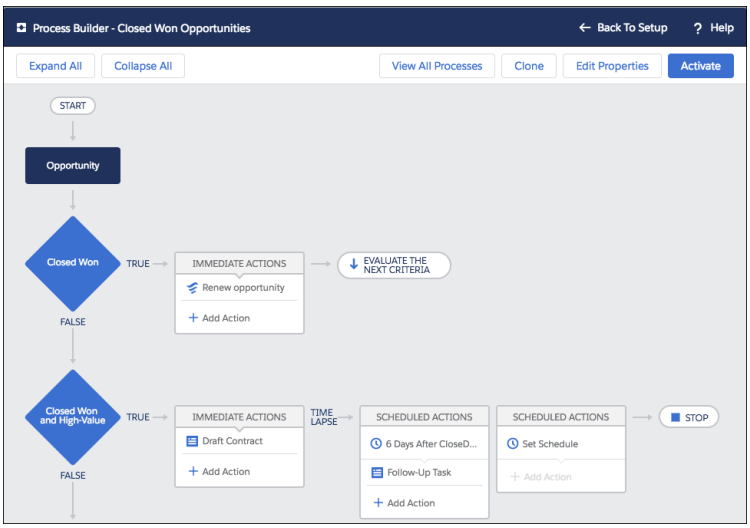


Figure 21 : L'interface du *Process Builder*

En effet, il n'existe que trois types d'éléments. Il y a l'objet autour duquel tourne le processus dans le rectangle bleu foncé, les critères dans les losanges bleu clair et les actions immédiates ou retardées dans les fenêtres blanches. Leur paramétrage est toujours similaire, sous réserve que les valeurs changent en fonction de l'élément concerné. Il suffit de remplir un formulaire avec les valeurs ou les variables à utiliser.

Par exemple, s'il s'agit d'effectuer des modifications à une entrée de la base de données, la fenêtre de paramétrage de l'action ressemble à celle de la Figure 22. On y définit son nom, le type d'entrée à modifier, les éventuelles conditions à satisfaire pour réaliser les modifications ainsi que les nouvelles valeurs à insérer.

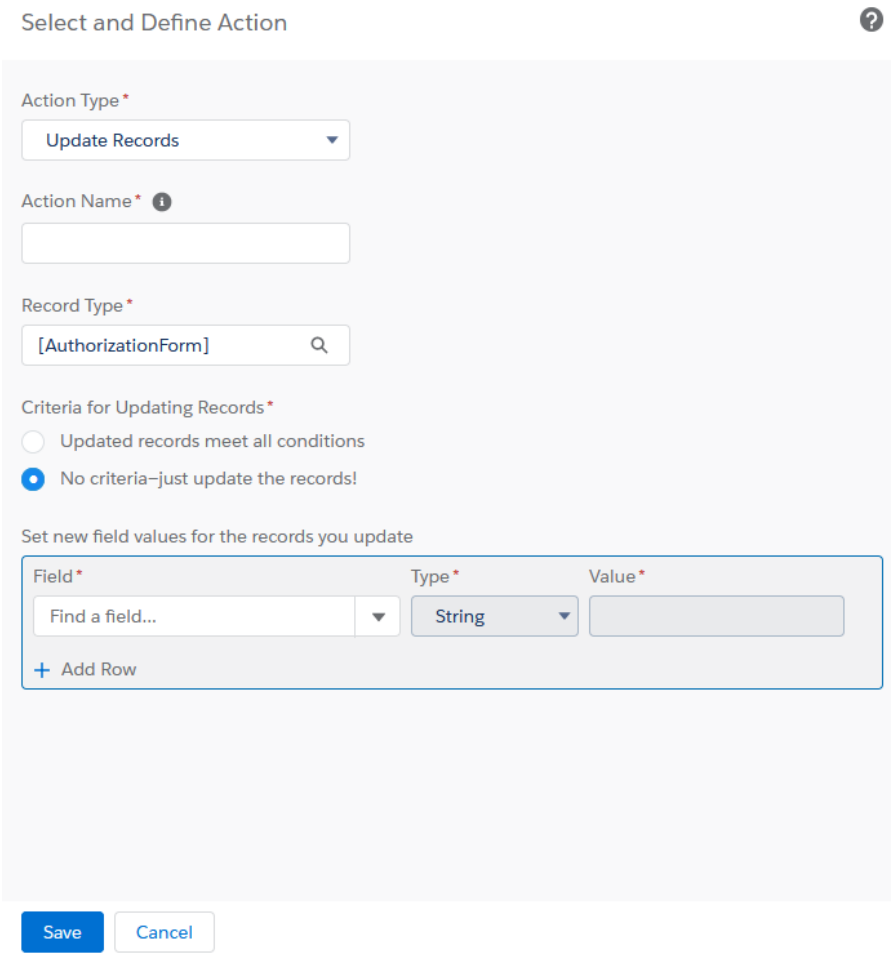


Figure 22 : Paramétrage d'une action de processus

Processus d'approbation

Au vu de l'importance des processus d'approbation et de leur utilisation courante dans les logiciels CRM, Salesforce a décidé de proposer un outil entièrement dédié à leur création. De fait, il n'est pas impossible de développer ce genre de processus via le *Process Builder*, mais c'est une tâche longue et fastidieuse. L'outil *Approval Process* a donc été ajouté à la plateforme afin de faciliter leur création. Il s'agit d'un dérivé du *Process Builder* dont l'interface est uniquement textuelle (Figure 25) et ne réalise que des tâches bien spécifiques.

Cependant, il n'est pas aussi trivial et intuitif à utiliser que les deux outils d'automatisation précédents. En effet, il est nécessaire de définir un certain nombre d'éléments et recommandé d'effectuer certaines vérifications avant de se lancer dans la création d'un tel processus. Il faut donc :

1. Consulter la checklist d'aide fournie[15]. Elle regroupe en 12 points les questions essentielles qu'il faut se poser avant de réaliser le processus : qu'est-ce qui le déclenche ? Qu'elles sont les étapes à effectuer ? Qui peut soumettre la demande ? Qui peut l'approuver ? Quelles sont les actions à effectuer lorsque la demande est approuvée ou rejetée ? Etc...
2. Hiérarchiser les utilisateurs au sein de l'application : au minimum deux rôles doivent être attribués, correspondant en gros à celui d'employé et de manager. Cela permet de déterminer qui peut soumettre une demande et qui peut l'approuver. Bien sûr, il est possible de créer autant de rôles que voulu, en définissant par exemple un Super Manager, un CEO et un CFO, qui ont chacun une implication différente dans le processus. On trouve un exemple de hiérarchie à la Figure 23.

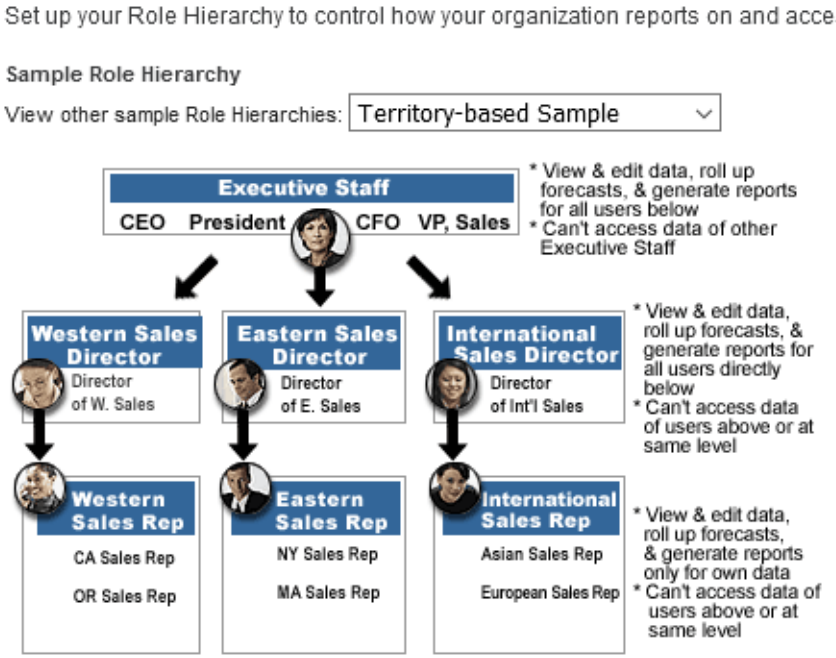


Figure 23 : Exemple de hiérarchie dans l'organisation

3. Créer un modèle d'Email. Puisqu'il est possible de notifier les utilisateurs par mails qu'une demande est arrivée (pour les approbateurs), puis acceptée ou refusée (pour les demandeurs), il est nécessaire de définir la forme de ces mails. Un outil dédié est intégré à Salesforce et permet d'ajouter des variables de la plateforme au texte. Très utile pour directement renseigner la demande, comme c'est le cas avec *{!Contact_Name}* de la Figure 24

[15] https://help.Salesforce.com/articleView?err=1&id=approvals_checklist.htm&type=5

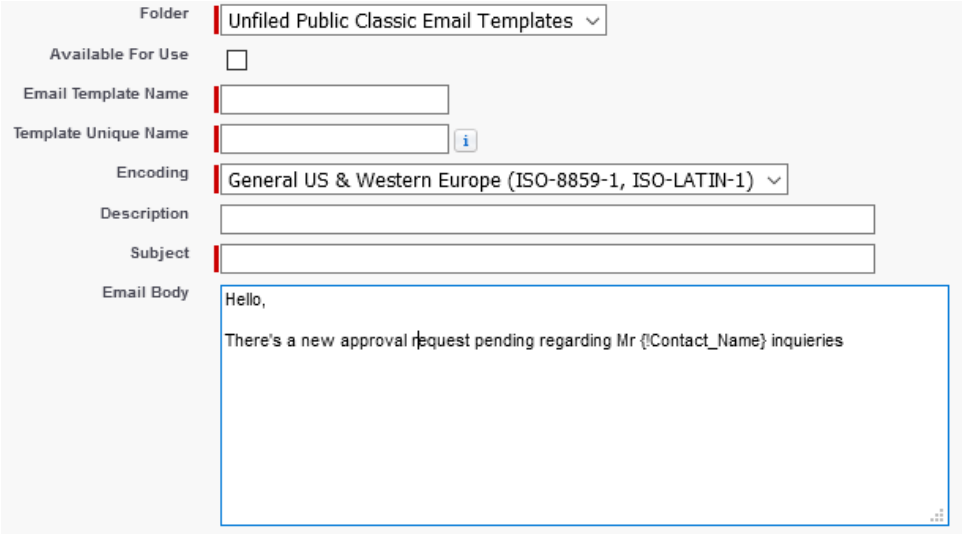


Figure 24 : Un modèle de mail

L'interface de développement que l'on retrouve à la Figure 25 est caractérisée par six zones. Dans la première on trouve les informations générales comme le nom du processus, les critères d'entrée, la liste des utilisateurs qui peuvent enclencher la demande d'approbation, le nom du modèle de mailing, etc.

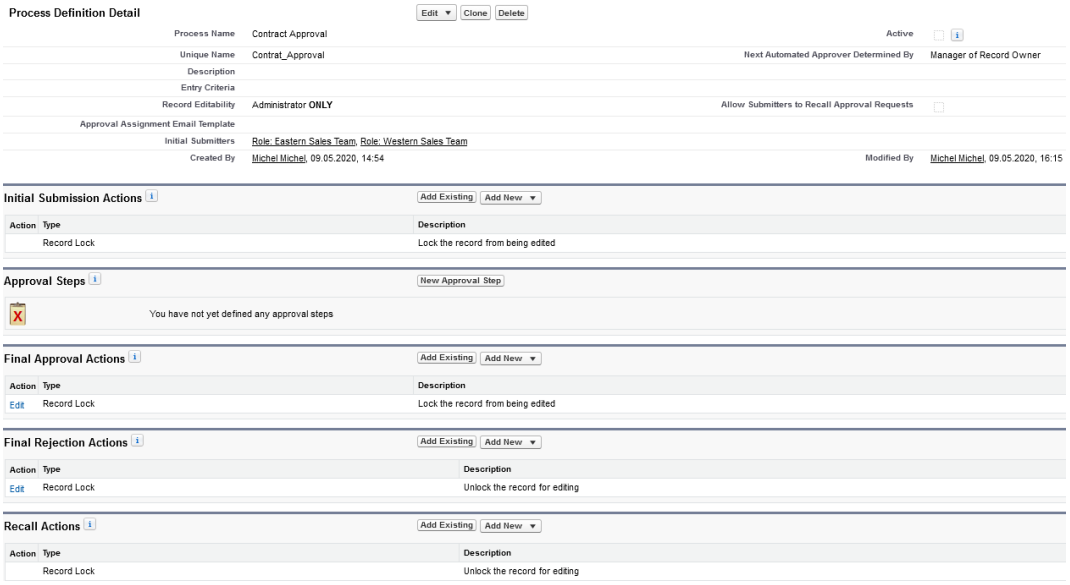


Figure 25 : Interface d'un *Approval Process*

Les cinq zones suivantes servent à décrire ce que fait le processus tout au long de son exécution. D'une part, il y a la zone avec les étapes du processus (les "Approval Steps") qui décrivent le chemin adopté par celui-ci avant d'arriver à son approbation. Par exemple, s'il s'agit d'approuver un rabais, la demande est envoyée au manager du magasin. En revanche, si le rabais dépasse 15%, la demande est aussi envoyée au Manager régional.

D'autre part, il y a les zones avec les actions à effectuer lors de la soumission, de l'approbation, du rejet et du rappel de la demande. À chacune de ces 4 étapes, il possible de :

1. Créer une nouvelle tâche dans l'agenda d'un utilisateur.
2. Envoyer un mail. L'adresse peut être personnalisée à la main, il n'est pas nécessaire qu'elle soit dans la base de données.
3. Mettre à jour un champ d'un objet selon certains critères.
4. Faire apparaître des notifications à certains utilisateurs.

Cet outil se révèle donc complet dans ce qu'il est capable de réaliser, mais également bien moins simple à utiliser que ses homologues d'automatisation. Il reste cependant le moyen le plus efficace et le plus rapide de construire des processus d'approbation sur la plateforme.

Vérification de format et formules

Il est parfois nécessaire de récupérer une information sous une certaine forme (numérique, alphabétique, booléenne, etc.) et remplissant certaines conditions (supérieures à 100, sans majuscules, uniquement 'vrai' ou 'faux', etc.).

Salesforce gère cela de plusieurs façons :

1. Lors de la création d'un champ dans la table de la base de données, il est nécessaire de sélectionner son type parmi une liste exhaustive : texte, monnaie, date, etc. (cf. chapitre 4.3.3). Ainsi, lors de l'ajout d'une entrée dans le système, il sera impossible de remplir de texte un champ prévu pour des nombres, par exemple.
2. Chaque objet de la base de données possède ses propres caractéristiques, affichage, boutons, champs, et surtout ses propres règles de validation (Figure 26**)**. Cela permet de s'assurer que chaque enregistrement d'un objet satisfait certaines conditions, comme le fait qu'un rabais ne puisse pas dépasser une certaine limite, par exemple. Pour utiliser cette fonctionnalité, il suffit de déterminer une formule logique et de spécifier le message d'erreur si elle n'est pas respectée.

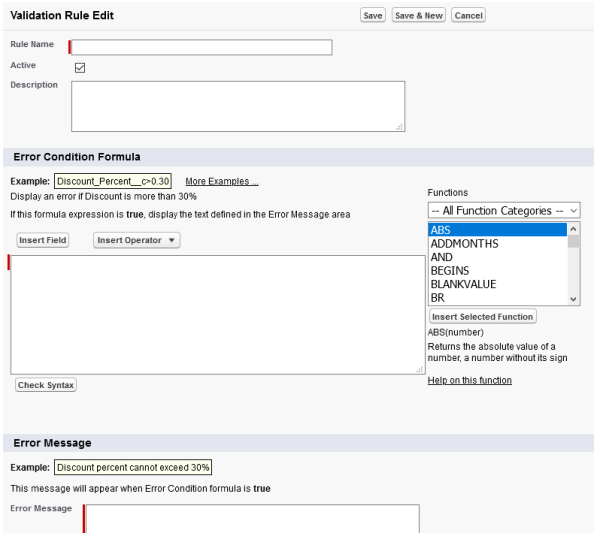


Figure 26 : Paramétrage d'une règle de validation

3. Lors de l'utilisation de *Flows*, les formules de validation d'input empêchent le processus de continuer si l'une des valeurs entrées est absurde.

Enfin, les formules constituent aussi un outils intéressant pour automatiser certaines tâches. Il s'agit de l'un des *types* attribuables à un champ quand on le définit dans un objet (cf. chapitre 4.3.3). En adéquation avec la formule logique qui y est définie, celui-ci utilise les valeurs présentes dans d'autres champs pour afficher un résultat.

Par exemple, cela de permet de calculer automatiquement son chiffre d'affaire en combinant les champs *{!Nombre_d'unités_vendues}* et *{!Prix_unités}*. Le résultat de ce calcul est ensuite affiché dans son propre champ dans l'objet.

Par ailleurs, cet outil ne se limite pas à manipuler des chiffres, comme on peut l'observer à la Figure 27**.** Cela permet de satisfaire les besoins en flexibilité lié à ce genre d'opération.

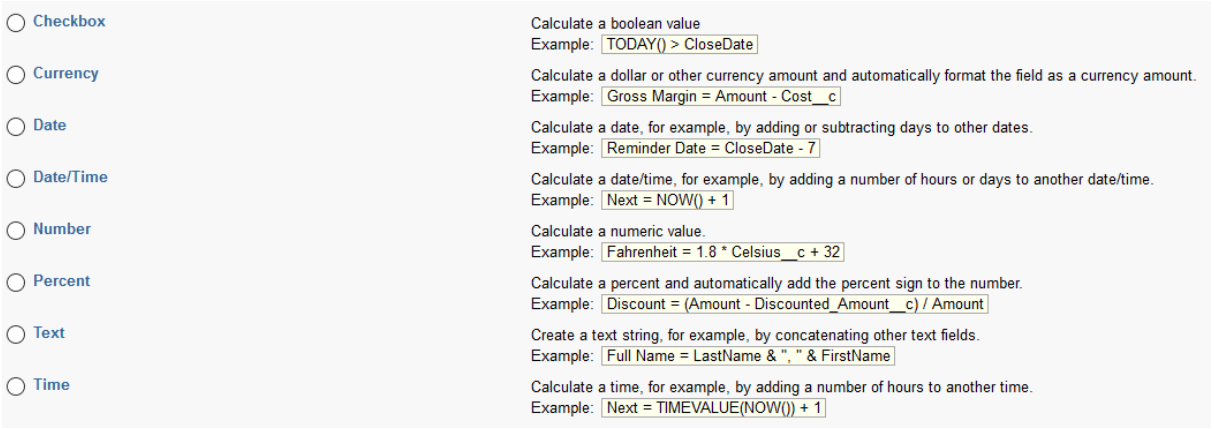


Figure 27 : Les types de formules

5
Application

L'application créée dans le cadre de ce projet a pour but de décrire au mieux les capacités de Salesforce et d'explorer au maximum ses outils. Ceux-ci sont très variés, peuvent demander beaucoup de paramétrage en amont de leur utilisation, et requièrent bien souvent un apprentissage préalable grâce à Trailhead.

C'est pourquoi ce chapitre ne traitera qu'une sélection des processus, workflow, règles de validation, raccourcis, choix d'interface utilisateur, etc. qui ont été implémentés. Il y a tellement de détails dont il serait possible de parler qu'il est nécessaire de se focaliser sur les éléments centraux les plus importants et les plus spécifiques de Salesforce.

De plus, l'application développée ne serait pas réellement utilisable par une entreprise. Seul un nombre limité de fonctionnalités a été implémenté pour permettre de réaliser certaines tâches précises, telles que créer un nouveau contrat avec tous ses prérequis et contraintes. C'est suffisant pour aborder un maximum d'outils sans pour autant passer des centaines d'heures à développer un produit dans sa totalité.

5.1 Interface utilisateur

5.1.1 Architecture de l'application

L'architecture adoptée est des plus classiques. Des pages sont accessibles grâce au menu central, et l'on peut interagir sur celles-ci pour effectuer certaines actions grâce à des boutons ou des flows prévus à cet effet. On a donc les pages suivantes :

- la page d'accueil (*Home page*) : la page centrale à partir de laquelle il est possible de contrôler la quasi-totalité de l'application
- six pages pour la gestion, la consultation et la modification des entrées de la base de données :
    - une page pour lister l'ensemble des biens immobiliers
    - deux pages relatives aux contrats de bail et de conciergerie
    - trois pages pour lister les dépendances liées aux appartements, aux maisons/villas et aux locaux commerciaux, respectivement.
- trois pages pour l'administration avec respectivement :

- un Chatter pour communiquer et poster des informations au sein de l'entreprise
- une liste offrant un suivi des tâches qui ont été assignées
- une liste, pour les managers, de l'ensemble des demandes d'approbation qui ont été émises, offrant la possibilité de les accepter ou de les refuser.

Home page

La construction de la *Home page* (Figure 28) a été réalisée de manière personnalisée grâce à l'outil *Page Builder* de Salesforce, qui permet de créer ses propres pages à la souris.

Tout en haut se trouve le *header*, commun à toutes les pages. Il contient le menu de navigation, une barre de recherche et les différents boutons de raccourcis (cf. Chapitre 4.3.1).

La page en elle-même est constituée de quatre zones. La première s'étend sur toute la largeur de l'écran, et comporte un accordéon contenant une fenêtre avec les tâches de la journée, un *dashboard* avec les événements de l'entreprise et son calendrier, ainsi qu'un *chat* pour publier des informations ou communiquer avec d'autres employés. Ces trois modules d'interaction améliorent l'expérience utilisateur et sont intégrés de base à Salesforce. Ils n'ont pas dû être créés de toutes pièces dans le cadre de ce projet, mais ont simplement été modifiés et insérés dans la page de façon à s'adapter à l'application développée.

Dans la seconde zone, en bas à gauche, se trouve le formulaire de *Flow* pour ajouter un bien immobilier à la base de données. La troisième zone, au centre, contient celui pour la création de contrats de bail et de contrats de conciergerie. La dernière zone à droite n'apparaît que sur l'écran des managers et contient la liste des demandes d'approbation en attente.

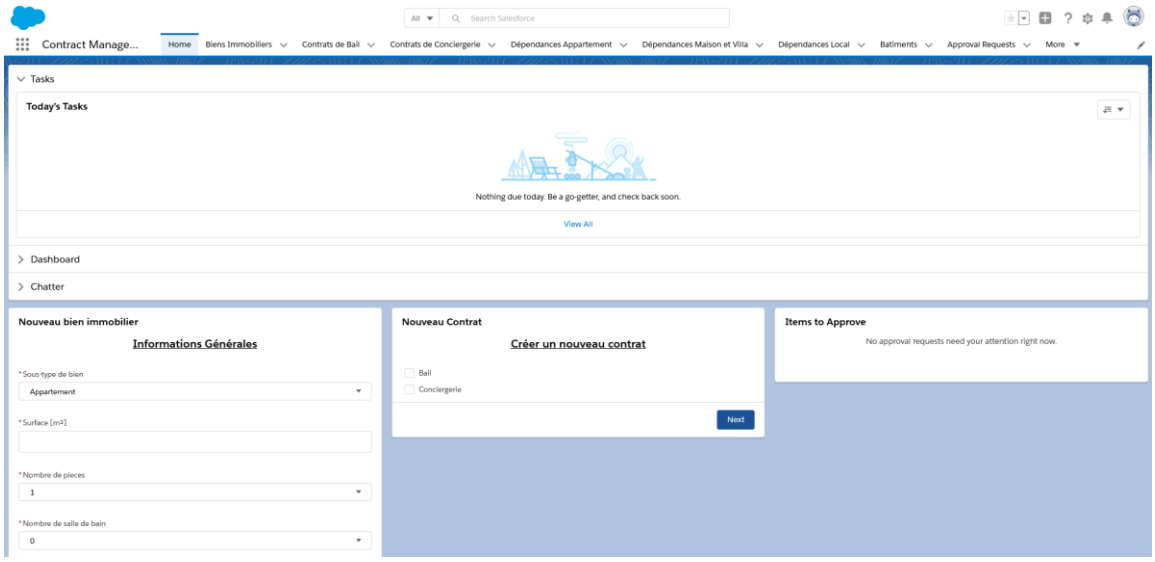


Figure 28 : Page d'accueil de l'application

Pages de gestion

Les six pages de gestion des objets de la base de données sont construites sur un même modèle, qu'il s'agit de décrire.

Premièrement, en cliquant sur leur onglet on obtient une liste des enregistrements (Figure 29). Un filtre personnalisé est appliqué pour faire ressortir les informations les plus pertinentes, telles que type de bien, loyer, surface etc. d'un bien immobilier. Des boutons pour ajouter et importer des données sont également disponibles mais uniquement pour les administrateurs, car il n'y a pas de règle de validation des informations lorsqu'on utilise ce moyen.

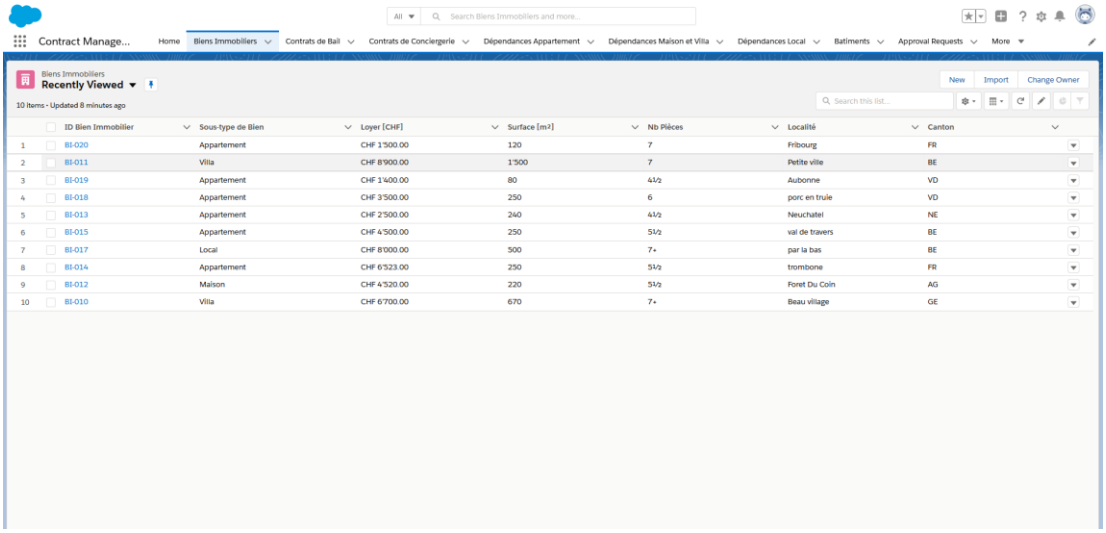


Figure 29 : La page listant les biens immobiliers

Cliquer sur l'ID d'un bien redirige sur une page qui récapitule ses informations (Figure 30)**.** En bas à gauche, on retrouve un formulaire de *Flow* pour ajouter une entrée de l'objet en question. Enfin, en bas à droite, on accède à un autre accordéon qui permet de saisir une nouvelle tâche ou un nouvel événement, de rédiger un mail ou encore de passer un appel, tout cela avec pour sujet l'objet de la base de données sur lequel on se trouve (bien immobilier, en l'occurrence).

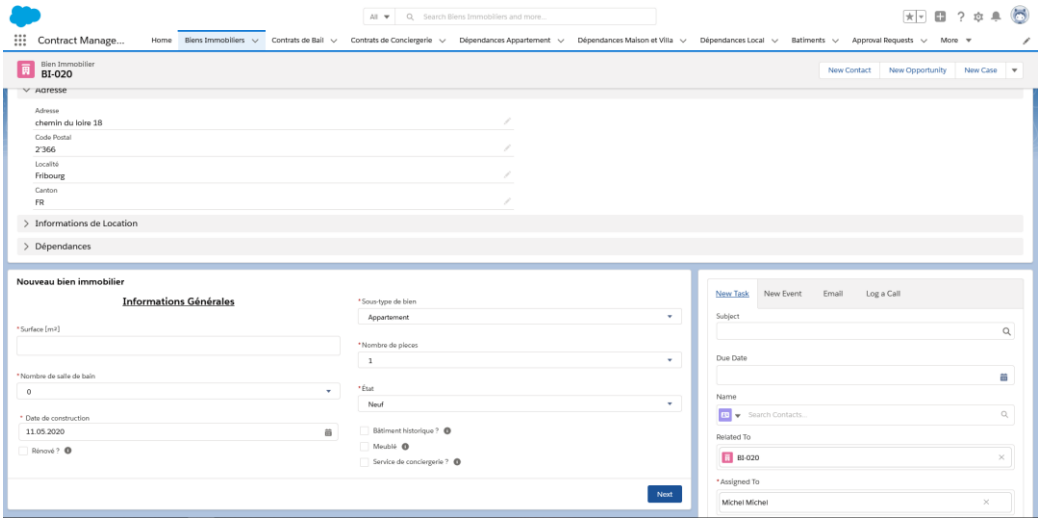


Figure 30 : La page de détail d'un bien immobilier

Pages d'administration

Les trois pages d'administration sont toutes différentes.

La page réservée au *chat* en direct est des plus classique et il n'est pas nécessaire de s'y attarder.

La page listant les demandes d'approbation en attente révèle un *lay-out* très semblable aux pages de gestion. Cliquer sur une demande renvoie vers une page regroupant les détails auquel le manager doit faire attention ainsi que des liens vers les objets concernés (Figure 31). Des commentaires peuvent être insérés dans la fenêtre de droite, tandis que les boutons pour approuver, rejeter ou réassigner la demande à un autre employé sont localisés dans le coin supérieur droit.

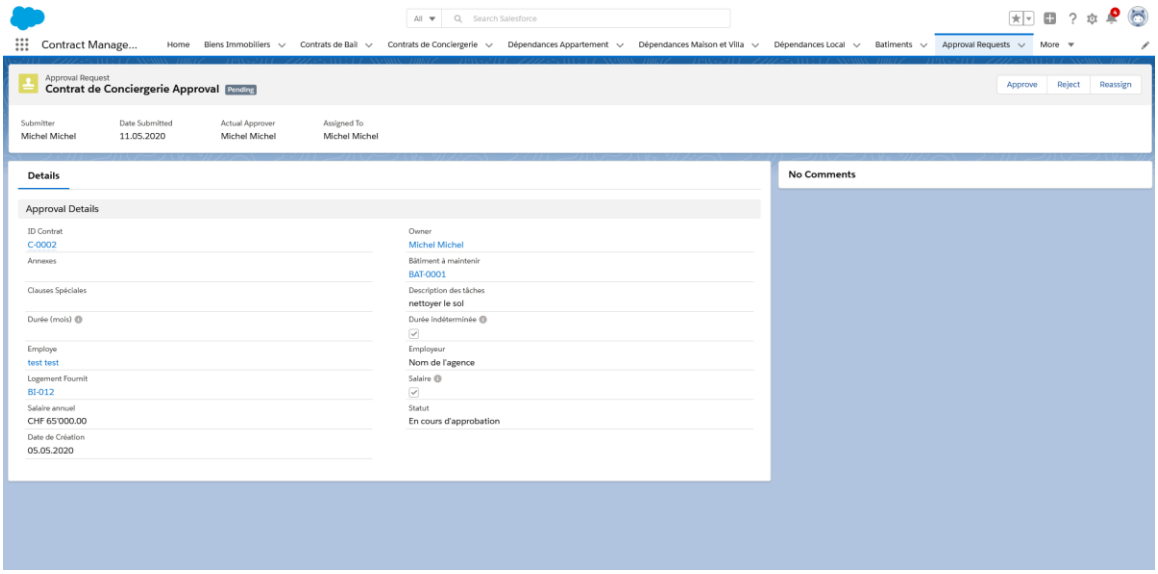


Figure 31 : Une demande d'approbation de contrat

Enfin, la page des tâches (Figure 32) est divisée en trois parties. À gauche, la liste des tâches à effectuer, que l'on peut trier ou filtrer selon divers critères : les plus urgentes, les plus récentes, etc.

La section supérieure comporte le nom de la tâche sélectionnée ainsi que les boutons d'administration qui servent, entre-autres, à marquer la tâche comme "complétée", ajouter des commentaires, changer la date butoir, créer une tâche secondaire, etc.

Dans la section centrale sont affichés les détails de la tâche, comme le nom de la personne à qui elle est assignée, l'objet auquel elle est liée, sa date butoir, etc. Cela permet de s'informer sur ce qu'il faut faire et jusqu'à quand en un coup d'œil.

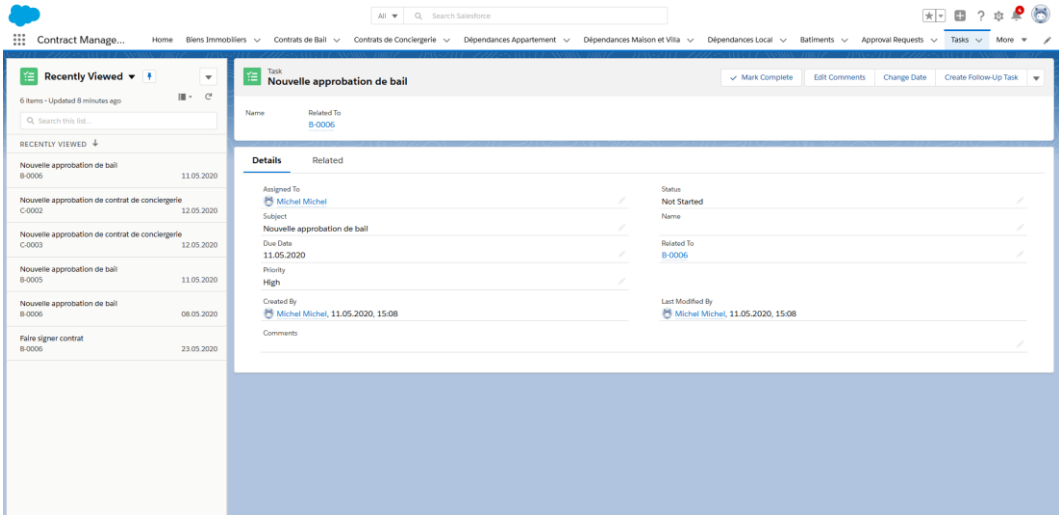


Figure 32 : La liste des tâches à effectuer

5.2 Base de données

5.2.1 Construction

La base de données est le point névralgique de l'application. En effet, c'est là que sont stockées toutes les informations de l'entreprise et que sont déterminés leur type, leurs compatibilités et leurs interactions. La Figure 33 est une capture d'écran de l'outil *schema builder* de Salesforce, qui permet de visualiser sa propre base de données.

Celle de l'application de gestion de contrats se profile ainsi :

- La table *Bien Immobilier* contient l'ensemble des informations communes à chaque bien, comme son adresse, sa surface, son loyer ou son nombre de pièces. Elle définit par un ID unique chacun des enregistrements qui y sont créés. De plus, elle reprend à deux reprises l'ID d'un contact grâce à des relations look-up. La première fois pour l'assignation d'un concierge et la deuxième fois pour définir l'occupant actuel du bien.
- Les tables *Dépendances Local, Dépendances Maison et Villa,* et *Dépendances Appartement* regroupent chacune les données spécifiques à leur type de bien. Les enregistrements de ces tables sont les seuls qui doivent être définis à l'aide de deux clés primaires. En effet, à cause de la relation « master-detail », il est nécessaire de renseigner l'ID du bien qu'ils détaillent en plus de leur ID unique.
- Les tables *Contrat de Bail* et *Contrat de Conciergerie* regroupent les informations des différents contrats du même nom. Chacun de leurs enregistrements possède sa propre clé primaire unique. Elles ont toutes deux des relations « look-up » qui leur permettent de :
    - décrire le bien qui fait l'objet d'une location
    - renseigner le nom du locataire ou du concierge à partir de ceux de la table *Contact*
    - montrer le logement offert au concierge en échange de son travail
    - lui assigner le bâtiment dont il faut s'occuper.
- La table *Contact* recèle toutes les données des personnes liées à l'entreprise, que ce soient ses employés, ses clients ou les concierges qu'elle mandate. Une relation « look-up » pointe vers cette même table pour déterminer les personnes tierces de contact.
- La table *Bâtiment* n'est pas complète puisqu'il s'agit d'une sorte de « triche ». En effet, elle est censée renseigner les biens immobiliers d'un bâtiment où peuvent être assignés des concierges. Cependant, implémenter cette fonctionnalité de façon dynamique prendrait un temps colossal et ne fait pas réellement partie du travail. Il s'agit dont d'un « bouche-trou » pour montrer qu'une relation existe entre les concierges et le bâtiment dont ils doivent s'occuper.

Il est à noter que certaines tables ne sont pas affichées. En effet, Salesforce en fournit automatiquement pour la gestion des tâches, des événements, des utilisateurs et des envois de formulaires Email. Chacune d'entre elles possède au moins deux relations avec toutes celles de la Figure 33, ce qui rend le schéma complétement illisible.

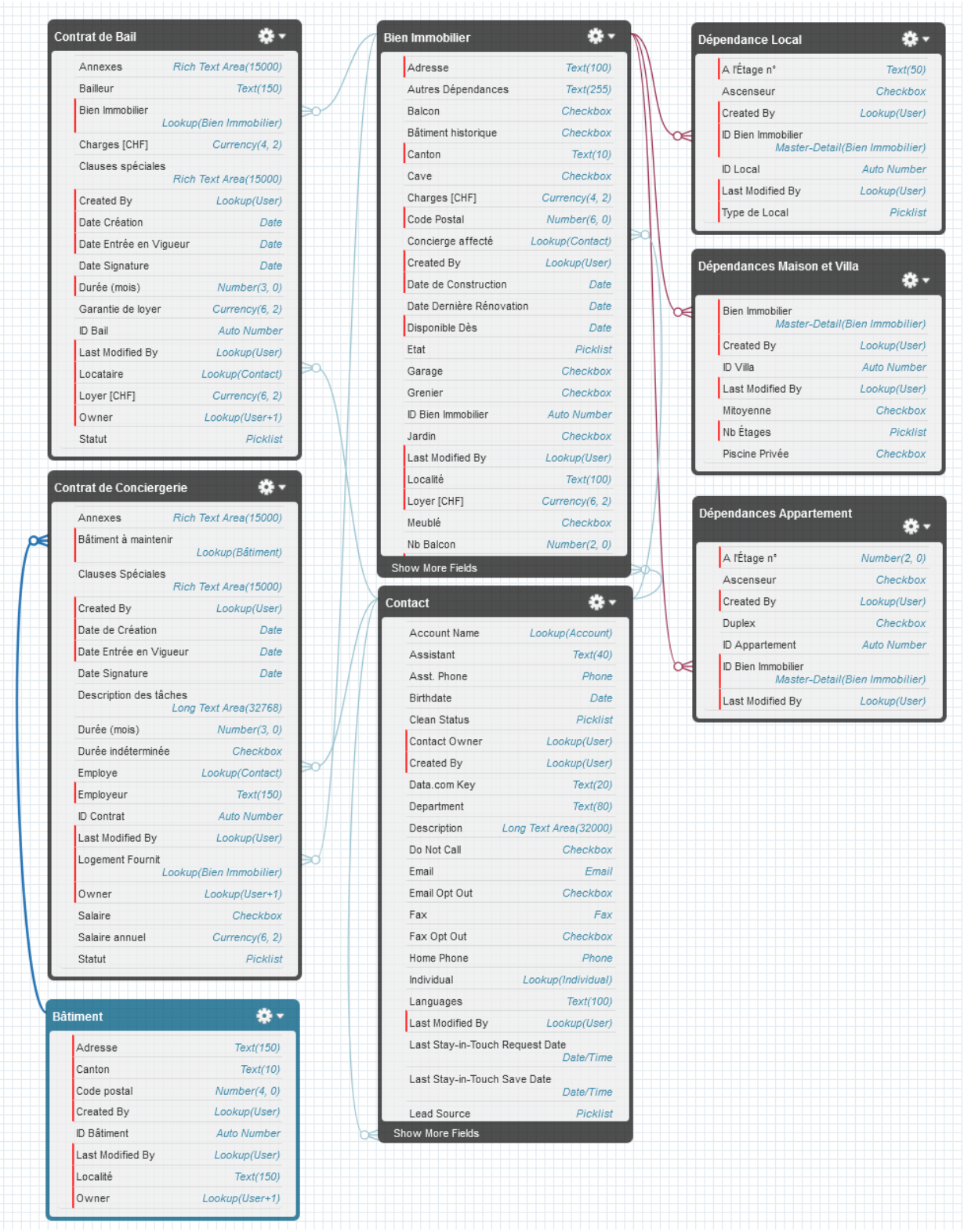


Figure 33 : La base de données de l'application

5.2.2 Limites de Salesforce

Contrairement aux systèmes de base de données relationnels plus flexibles comme MySQL, le fait de n'avoir à sa disposition que deux types de relations (lookup et master-detail) entraîne certaines complications, qu'il s'agit d'aborder. C'est pourquoi la base de données de l'application de gestion de contrat a été construite de façon à explorer deux alternatives d'implémentation à la fois, pour mettre en avant leurs avantages et leurs inconvénients. En effet, d'un côté on respecte l'unicité des données pour faciliter les interactions au sein du système, mais cela se fait au prix d'une utilisation générale moins intuitive. De l'autre, on crée des doublons de données, ce qui rend l'implémentation de processus plus complexe mais simplifie la vie des utilisateurs finaux.

Alternative n°1 : L'unicité des données au détriment de l'intuitivité d'utilisation

Le premier cas concerne l'implémentation de l'objet parent *Bien Immobilier* et de ses enfants, les objets *Dépendances (x)*. En effet, on retrouve dans cette première table l'ensemble des informations communes à chaque bien immobilier, comme l'adresse, le nombre de pièces, le loyer, potentiellement une place de parc, une cave, etc.

On décide ensuite de détailler les dépendances liées au type de bien. Par exemple, une maison/villa peut être mitoyenne, ce qui n'est pas le cas d'un appartement ou d'un local. Au même titre, un appartement peut être en duplex, mais pas un local ou une maison/villa.

Le point essentiel ici réside dans la relation "master-detail" entre un bien et ses dépendances particulières. En centralisant dans une table toutes les données communes et en détaillant dans d'autres les données spécifiques, on arrive à éviter un triplet d'informations identiques. Il n'y a pas besoin de faire la distinction entre l'adresse d'un *appartement* et l'adresse d'un *local*, par exemple ; c'est simplement l'adresse d'un *bien immobilier*. En revanche, à cause de la façon dont est implémenté l'outil de base de données, il n'est possible d'accéder à l'objet parent que depuis l'objet enfant, mais pas l'inverse.

Concrètement, cela veut dire qu'à la Figure 34 il est possible à partir de l'onglet *Dépendances Maison et Villa* d'être redirigé vers l'onglet *Bien Immobilier* en cliquant sur son ID (BI-011, par exemple), mais aucun champ n'existe dans ce dernier pour faire l'opération inverse. Impossible donc de consulter les dépendances particulières d'un bien à partir de l'onglet *Bien Immobilier*, ce qui s'avère loin d'être pratique.

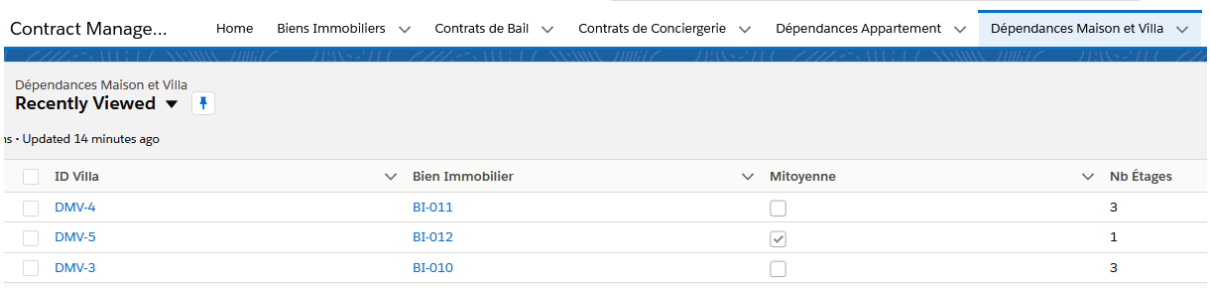


Figure 34 : La page des dépendances Maison et Villa

Pour y parvenir, cela nécessiterait l'ajout d'une relation "lookup", mais Salesforce ne permet pas de le faire pour différents types d'objets au sein du même champ. Ce qui veut dire que trois champs seraient alors nécessaires : un pour recevoir l'ID des appartements, un autre pour celui des maisons/villas et un dernier pour celui des locaux. Un seul parmi les trois serait alors renseigné à la fois, laissant ainsi les deux autres vides mais néanmoins visibles et modifiables dans l'interface utilisateur. Cela s'avère peu judicieux au vu de la propension des utilisateurs à perturber les systèmes par des manipulations imprévues.

Un travail colossal d'implémentation est alors obligatoire pour contourner ce problème, demandant entre autres la création de trois interfaces utilisateurs différentes qui seront affichées en fonction du type de bien, sans parler des multiples processus nécessaires pour effectuer la rocade. Cela requiert des compétences bien au-dessus de celles d'un développeur junior sur la plateforme et n'a donc pas pu être réalisé pour l'application de gestion de contrat.

Concrètement, la façon dont l'application est implémentée actuellement demande d'un employé de l'agence qui veut connaître les dépendances spéciales d'une villa de d'abord noter l'ID présent dans la table *Bien Immobilier* pour ensuite l'insérer dans la barre de recherche de la table *Dépendances Maison et Villa*. Ce n'est pas une opération des plus efficace, ni des plus intuitives, mais cela fonctionne et permet de contourner le problème grâce à une manipulation de l'utilisateur plutôt que du système.

Alternative 2 : la facilité d'utilisation au coût de la multiplicité des données

Le deuxième cas concerne l'implémentation des objets *Contrats de Bail* et *Contrats de conciergerie*. En effet, à l'instar des biens immobiliers, une entité *Contrat* aurait pu être créée pour regrouper les informations communes à ceux-ci, puisque chacun possède entre autres une date de création, de signature et d'entrée en vigueur, par exemple. Mais dans le cas présent il a été décidé de faire différemment et de considérer les objets comme deux entités distinctes.

Cela veut dire que deux variables semblables en tous points doivent être créées pour chaque information commune ; une pratique qui n'est pas encouragée en gestion de base de données. Par contre, cela permet de stocker les informations spécifiques au type de contrat directement dans l'objet. Puisqu'il n'y a pas de données « à l'extérieur », un utilisateur a donc accès à leur intégralité depuis la page *Contrats de Bail* sans devoir effectuer une recherche par ID dans une autre table. Par exemple, le loyer qui est une dépendance particulière des contrats de bail est directement disponible, comme on peut le constater à la Figure 35.

Durée (mois)	Locataire	Loyer [CHF]
60	Babara Levy	CHF 3'500.00
60	test test	CHF 8'000.00
50	Alice Black	CHF 4'500.00
60	Alice Black	CHF 6'523.00
12	test test	CHF 1.00

Figure 35 : La page des contrats de bail

Cependant, l'existence d'une multitude de variables en doublon complique l'implémentation des processus du système, comme on peut l'observer avec le workflow de création de contrat (Figure 42). En effet, il est alors nécessaire de prévoir deux branches, chacune avec leurs propres ressources, validations d'input, interfaces, etc. Ce qui a été mis en place ici n'est qu'un processus de complexité moyenne, avec seulement deux objets différents. Le travail à effectuer pour gérer les sources d'erreurs, les validations d'input et la gestion des contraintes des données augmente en fonction du nombre d'objets impliqués. S'il y avait eu 30 types de contrat différents, ce type d'implémentation n'aurait pas pu être réalisable.

Salesforce a beau se targuer d'être facile d'utilisation avec des outils adaptés et didactiques, son système de base de données n'est pas aussi flexible qu'on pourrait s'y attendre. La limitation des relations à seulement deux types entraîne des complications lorsque l'on cherche à implémenter certaines fonctionnalités très spécifiques. Ce genre de défaut est facilement contournable avec un systèmes de base de données géré en code, comme MySQL, mais moins lorsque l'on utilise des outils préconçus. Leur niveau d'abstraction supérieur implique nécessairement des concessions en termes de fonctionnalité et de souplesse.

5.3 Processus

De très nombreux petits processus ont été développés pour l'application de gestion de contrat à l'aide du *Process Builder,* afin d'effectuer des tâches en arrière-plan. Certains ont pour objectif de gérer les données dans la base tandis que les autres traitent les processus d'approbation (Chapitre 5.4).

Ils sont tous implémentés d'une façon similaire, aussi n'est-il pas nécessaire de les passer tous en revue puisque le principe reste le même à chaque fois.

Par exemple, lorsqu'un contrat de bail entre en vigueur, il est nécessaire de modifier le statut d'occupation du bien immobilier dont il fait l'objet, qui passe de « disponible » à la location à « occupé » par un locataire (Figure 36).

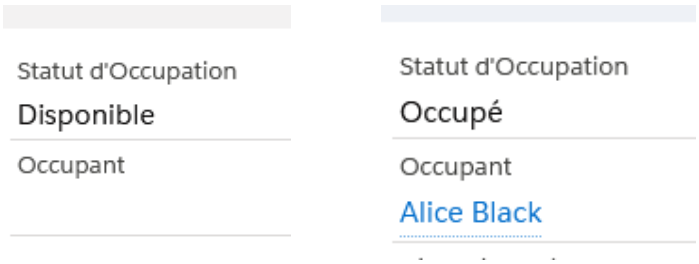


Figure 36 : Variation des champs d'occupation (avant – après)

Au lieu de devoir effectuer manuellement ces modifications, on développe un *Process* qui surveille les changements apportés à l'objet "Contrats de bail" pour le faire (Figure 37). Ainsi, le système reconnaît lorsque le statut du contrat passe à la valeur "En vigueur" et va modifier les entrées du bien immobilier. En l'occurrence, il s'agit des champs *Occupant* et *Statut d'occupation*, qui prennent respectivement le nom du locataire (tiré directement du contrat grâce à une relation *look-up*) et la valeur "Occupé".

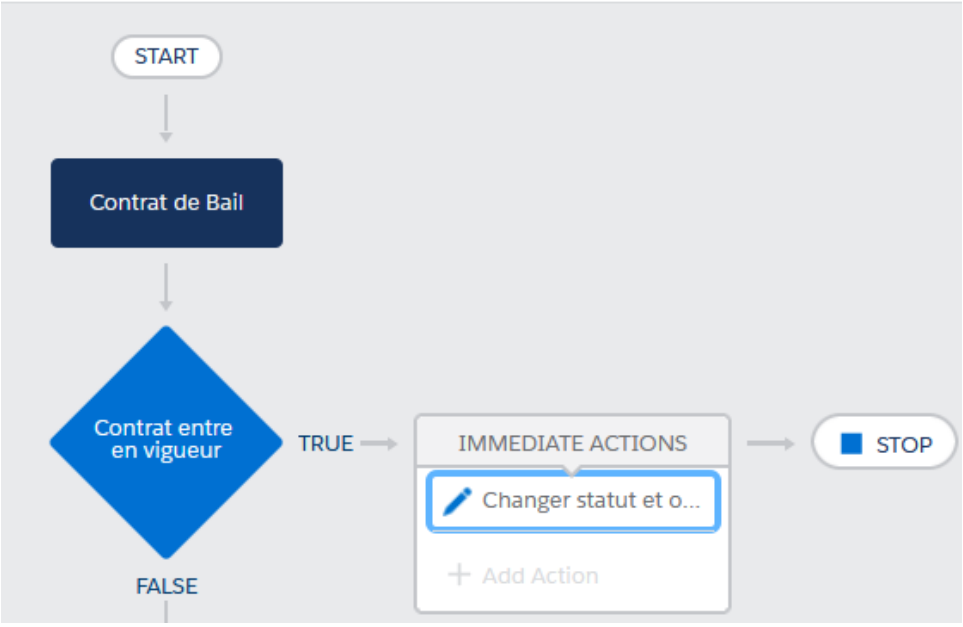


Figure 37 : Le processus surveillant le statut des contrats de bail

Les processus de l'application contiennent tous trois étapes :

1. On indique l'objet que l'on veut surveiller (rectangle bleu foncé). Pour l'exemple actuel, il s'agit de celui des contrats de bail.
2. On indique les conditions pour effectuer certaines tâches (losange bleu clair), par exemple si le contrat entre en vigueur.
3. On indique les actions à effectuer, comme changer la valeur de certains champs (rectangle blanc).

Ces trois étapes sont donc la base de tous les processus de l'application, que ce soit pour envoyer une demande d'approbation lorsqu'un contrat vient d'être enregistré, ajouter les informations d'un concierge à un bien immobilier quand il est recruté ou encore retirer de la liste des biens disponibles à la location un bien qui entre en rénovation ou n'est pas en état d'être loué.

On pourrait bien entendu créer des processus plus complexes comportant davantage d'étapes liées à une multitude d'objets, pour effectuer des actions en simultané, ou déclenchées après un certain délai. Cependant, cela n'est pas nécessaire pour se rendre compte de la grande utilité de cet outil, dont les possibilités d'automatisation sont gigantesques.

5.4 Processus d'approbation

Deux processus d'approbation ont été développés et s'enclenchent lorsqu'un contrat de bail ou un contrat de conciergerie est créé et complété par un employé. Il n'est pas nécessaire de les décortiquer tous deux, puisqu'ils sont à peu de choses près semblables. Comme l'outil *Approval Processes* ne permet de créer des processus que pour un objet en particulier, il est impossible de les réutiliser et donc nécessaire de tout redéfinir (hormis les modèles de mailing et la hiérarchie de l'organisation).

La Figure 38 contient plusieurs informations générales liées à la description du processus :

1. L'entrée dans le processus ne se réalise que si le champ « Statut » de l'objet en question (Contrat de bail) a la valeur « En attente d'approbation ».
2. Seuls les administrateurs ont la possibilité de modifier des champs après que la demande a été effectuée et tant qu'une décision n'a pas été prise.
3. Le modèle de mailing est celui de "Contrat à approuver", qui a été créé de façon personnalisée pour y intégrer des informations utiles à ce type de demande.
4. Seul le créateur du contrat, en sa qualité de responsable, peut en solliciter l'approbation, lorsque les données correspondent aux exigences fixées par les managers.
5. L'auteur d'une demande d'approbation n'est pas autorisé à la retirer ultérieurement, en raison d'un paramètre de blocage (non mentionné sur la Figure 38). Les approbateurs la verront donc forcément.

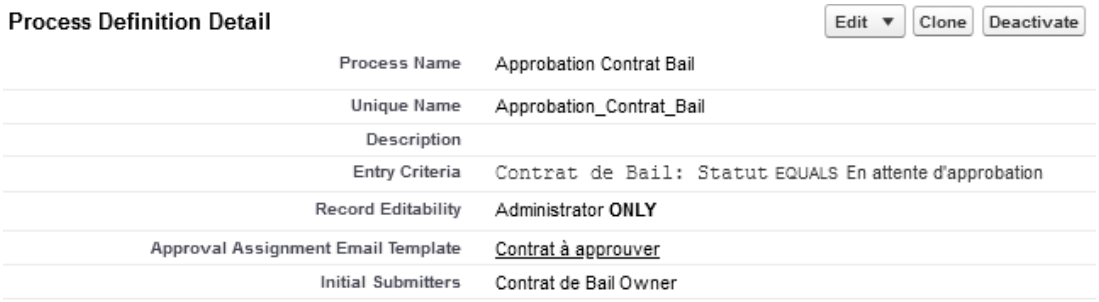


Figure 38 : Informations générales du processus d'approbation

Les actions réalisées par les deux processus sont semblables, et on les retrouve en détail à la Figure 39. Concrètement, ces descriptions signifient que dès lors qu'une demande est envoyée, le contrat ne peut plus être modifié et son statut passe à "En attente d'approbation". De plus, une tâche est créée dans l'agenda des approbateurs pour les informer qu'une nouvelle demande a été émise.

Si la demande est approuvée, le statut du contrat change à "Approuvé" et l'employé se voit assigner de le faire signer pour qu'il entre en vigueur. Si la demande est rejetée, le statut du contrat change en "Refusé" et l'employé se voit assigner de le modifier, après quoi il devra renouveler sa demande à l'aide du même processus d'approbation.

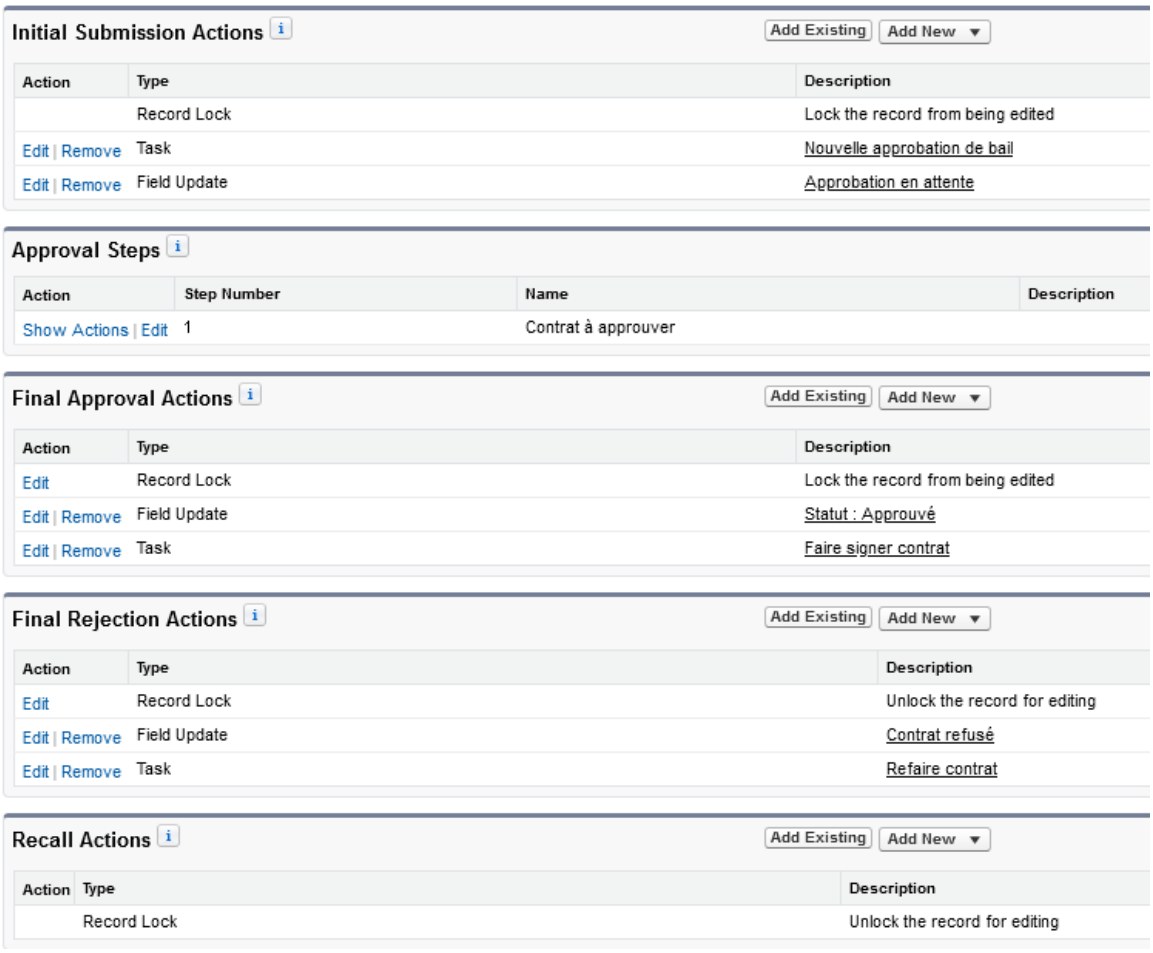


Figure 39 : Détails et étapes du processus d'approbation

5.5 Workflows

Deux workflows centraux ont été implémentés pour l'application web. L'un guide l'utilisateur lors de l'ajout d'une propriété dans la base de données, tandis que l'autre l'aide lors de la création de contrats.

Conformément aux particularités liées à la base de données, ils ont chacun un mode de fonctionnement différent, mais l'implémentation reste semblable. En effet, l'objectif commun est relativement simple : faire en sorte que l'utilisateur saisisse les bonnes informations dans les champs d'un objet pour que sa création soit acceptée par le système. Les règles de validation sont donc omniprésentes à chaque écran.

Ajout d'un bien immobilier

Le premier workflow décrit par le schéma de la Figure 40 est celui permettant d'ajouter un bien immobilier dans le système.

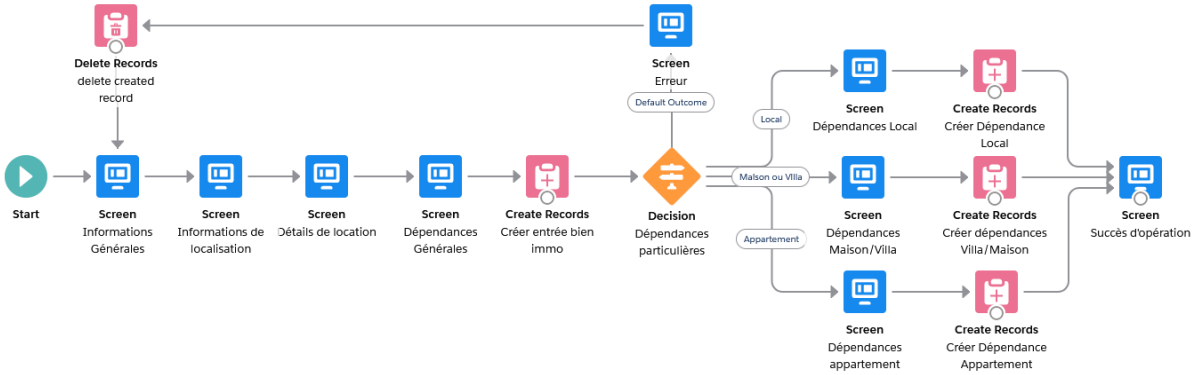


Figure 40 : Schéma du workflow des biens immobiliers

La première partie, avant d'atteindre le nœud de décision (losange orange), demande à l'utilisateur de compléter tous les champs de l'objet *Bien Immobilier*, c'est-à-dire les informations communes à chaque bien. Elle est divisée en 5 étapes :

- quatre écrans, lesquels pourraient très bien n'en faire qu'un, mais qui sont fractionnés pour plus de confort d'utilisation, en raison du grand nombre d'informations à renseigner. Par exemple, le deuxième écran comporte les champs de localisation du bien (Figure 41)
- une création d'objet.

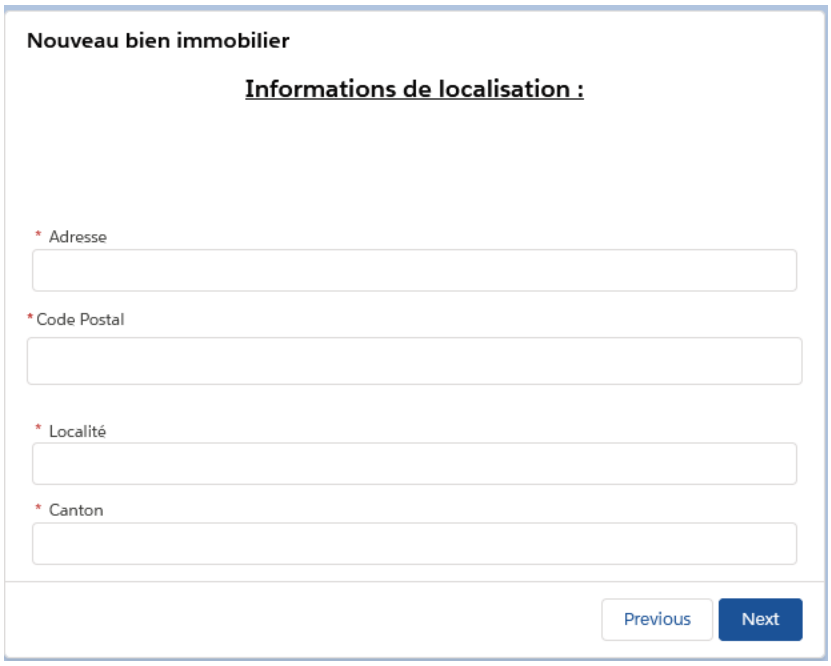


Figure 41 : Formulaire de l'écran n°2

La création du bien immobilier a lieu avant le nœud de décision puisqu'on aura besoin de son ID pour l'insérer à l'étape suivante dans la relation "master-detail" des dépendances particulières. Elle est placée ici afin de ne pas avoir à l'ajouter après chaque alternative de décision.

La seconde partie commence avec le nœud de décision, par lequel on détermine la catégorie du bien immobilier, avant de poursuivre par la saisie de ses dépendances spécifiques. Comme les champs sont différents selon qu'il s'agit d'un Local, d'une Maison/Villa ou d'un Appartement, il faut trois filières distinctes qui convergent en un écran commun marquant le succès de la création. L'annulation de la création reste possible avec l'option *Default outcome*.

Création d'un nouveau contrat

Le deuxième workflow décrit par la Figure 42 est celui permettant de guider l'utilisateur dans la création d'un contrat.

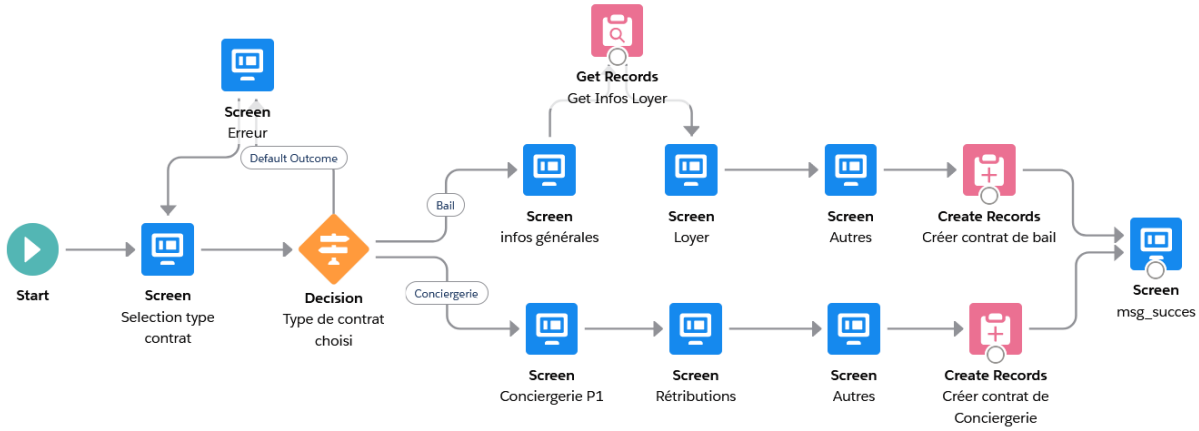


Figure 42 : Schéma du workflow des contrats

Puisque les relations entre biens immobiliers et contrats sont différentes dans la base de données, le workflow traitant de ces derniers est façonné d'une autre manière. Ainsi, l'utilisateur choisit le type de contrat au premier écran, ce qui détermine la suite du processus.

En effet, s'il s'agit d'un contrat de conciergerie (branche du bas), un enchaînement de trois écrans sert à récupérer ses informations. On crée ensuite l'objet pour finir sur un message de succès de l'opération.

Cependant, s'il s'agit d'un contrat de bail, il est d'abord demandé à l'utilisateur de renseigner des informations relatives au bien immobilier qui l'intéresse. De la sorte, l'application identifie les valeurs de loyer correspondantes et les reprend automatiquement dans l'écran suivant. Il n'est donc pas nécessaire de les saisir soi-même, ce qui évite le risque d'erreur. On crée ensuite l'objet pour finir sur un message de succès de l'opération.

Paramétrage des éléments d'écran

Chaque écran apparaissant dans une fenêtre de l'application peut être paramétré pour fournir une expérience agréable à l'utilisateur en fonction de ses choix ou des informations rentrées au préalable. Ainsi, il est possible de contrôler la visibilité des éléments apparaissant ou de valider les inputs selon une formule. C'est ce qui a été appliqué à chaque composant des *flows* de cette application, mais les passer tous en revue serait redondant. Aussi ne sont abordés qu'un exemple de chacun.

Le premier concerne la validation d'input. Pour chaque nouveau contrat, sa date de création doit être indiquée. Par défaut, la date du jour actuel est proposée, mais elle peut être modifiée s'il s'agit de reprendre un contrat plus ancien. Les dates possibles sont donc la date du jour ou une date antérieure. En revanche il doit être impossible pour l'utilisateur de saisir une date future.

C'est ce que vérifie la formule décrite à la Figure 43. Concrètement, un input rentré dans le champ *{!B_Date_Creation}* doit correspondre à une date antérieure ou présente. Sinon, un message d'erreur personnalisé est affiché à l'écran et le processus est suspendu tant que l'erreur n'est pas corrigée.

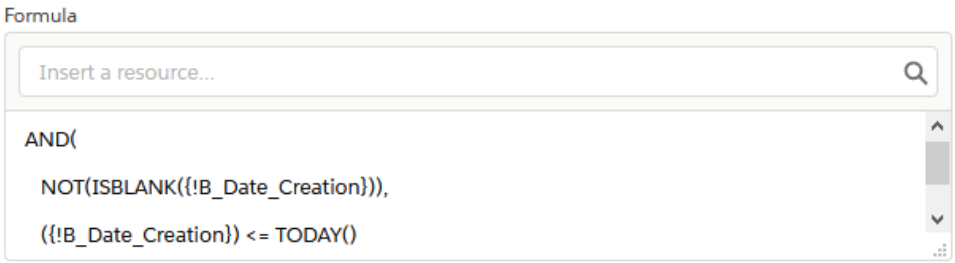


Figure 43 : Formule de validation d'input

Le second exemple concerne la visibilité d'un élément en fonction d'un choix effectué par l'utilisateur. Lors de la saisie des dépendances générales d'un bien immobilier, ce dernier doit notamment renseigner si un balcon y est rattaché ou non, à l'aide d'une *checkbox* dont l'output est "True" si elle est cochée, "False" dans le cas contraire.

Ainsi, le champ "Superficie du balcon [m²]" a été programmé pour devenir visible lorsque la formule de la Figure 44 est respectée, c'est à dire lorsque la *checkbox {!Balcon}* est cochée. Cela rend le formulaire plus compact et surtout garantit que l'utilisateur n'a accès qu'aux champs pertinents pour chaque bien.

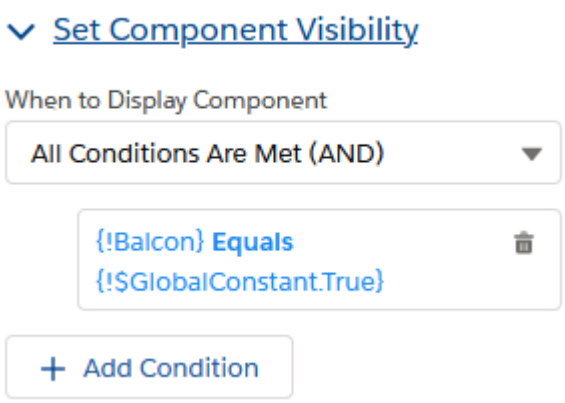


Figure 44 : Conditions de visibilité

5.6 Prochaine étape de développement

Pour la réalisation de ce projet, une hiérarchie des employés a été créée au sein de l'organisation. Mais elle n'est pas exploitée au maximum de ses possibilités, puisqu'elle n'est impliquée que dans les processus d'automatisation, et plus particulièrement ceux d'approbation.

En effet, Salesforce propose de nombreuses fonctionnalités supplémentaires liées à cette hiérarchie, dont notamment celle de confidentialité des informations. Concrètement, cela signifie qu'en fonction de son rôle, l'utilisateur ne peut avoir accès qu'à certains onglets, certains champs, certains outils, etc. On pourrait donc implémenter dans ce but cette fonctionnalité, dans un développement plus poussé de l'application.

Par exemple, en se calquant sur la hiérarchie existante, on peut envisager les restrictions suivantes :

1. Les clients ne peuvent que se créer un compte, consulter les biens immobiliers déjà présents dans la base de données, témoigner de leur intérêt pour un bien qu'ils souhaitent louer, ou encore y ajouter leur propre bien (soumis à approbation) s'ils souhaitent le mettre en location.
2. Les concierges ne peuvent que s'inscrire dans le système, consulter les offres d'emploi et postuler pour s'occuper d'un bâtiment en particulier.
3. Les employés administratifs ont un compte fourni par l'agence immobilière, grâce auquel ils peuvent créer ou modifier un contrat et le soumettre pour approbation. Ils ont accès à l'ensemble des biens immobiliers, à la liste des contrats créés, peuvent communiquer via le chat intégré et organiser leur travail grâce au module de tâches de l'application.
4. Les managers ont les mêmes permissions que les employés, auxquelles s'ajoute la capacité d'approuver ou de rejeter les contrats. De plus, ils sont les seuls à pouvoir effacer un contrat ou un bien immobilier de la base de données.
5. Les administrateurs ont accès à toutes les fonctionnalités précédemment énumérée (pour effectuer des tests lors de leur développement, le support, etc.) ainsi qu'à l'interface de développement.

Il ne s'agit là que d'une liste de restrictions totalement arbitraires. Elle peut bien sûr changer en fonction de l'entreprise, du contexte d'utilisation de l'application, etc. Mais cela permet de se représenter les capacités de la plateforme à ce niveau.

6
Conclusion

Ce travail avait pour objectif de démontrer les capacités de Salesforce pour créer une application web. Pour ce faire, une application de gestion de contrats d'agences immobilières a été développée avec les outils de la plateforme.

Des informations théoriques sur les ERP, les CRM, le Cloud et Salesforce ont d'abord permis de mieux comprendre les prérequis d'un tel projet. Il a fallu ensuite aborder le concept de l'application, les études de marché menées à ce sujet ainsi que les particularités de la loi suisse concernant les contrats d'agence immobilière. Puis, ont été détaillés le fonctionnement des différents outils de Salesforce. Enfin, l'application a été décrite de manière détaillée pour mettre en évidence les points faibles et les points forts de la plateforme.

Celle-ci n'est pas exempte de défauts. D'une part, il s'agit d'une entreprise américaine, pays dont les lois sur la protection des données ne sont de loin pas aussi strictes qu'en Suisse, ce qui en rend l'utilisation délicate lorsqu'il est nécessaire de manipuler des informations sensibles. D'autre part, sa base de données ne peut être construite qu'avec deux types de relations, ce qui en fait un outil peu flexible. Cas échéant, il faut consacrer beaucoup de temps à développer des alternatives pour contourner ce défaut.

Cependant, Salesforce reste un excellent outil dont les capacités sont phénoménales. Il est possible d'y créer de nombreux processus, d'automatiser l'entièreté de son application et d'y développer très facilement n'importe quel type de logiciel. Il peut être ensuite publié en un clic sur le marché des applications de la plateforme, AppExchange.com, afin de le commercialiser.

De plus, le site Trailhead.com permet d'apprendre aisément le mode de fonctionnement de tous les outils et incite la communauté à toujours s'améliorer en ajoutant régulièrement de nouveaux badges et de nouveaux tutoriels.

Cela fait de Salesforce l'un des meilleurs moyens actuels pour acquérir de l'expérience en développement d'application et des concepts qui y sont liés. Toutefois, ce que l'on y développe n'a pas de réel potentiel commercial sur le marché suisse.

Sources

[1] Salesforce.
https://www.salesforce.com/eu/ (accédé le 5 mai 2020)

[2] Salesforce, théorie des bases de données.
https://developer.Salesforce.com/page/FR:Tout_savoir_sur_la_base_de_donn%C3%A9es_de_Force.com (accédé le 12 avril 2020)

[3] Salesforce, définition de Trailhead.
https://help.Salesforce.com/articleView?id=mth_what_is_trailhead.htm&type=5 (accédé le 10 janvier 2020)

[4] Trailhead, liste des modules.
https://trailhead.salesforce.com/fr/modules (accédé le 15 février 2020)

[5] Trailhead, théorie des workflow.
https://trailhead.salesforce.com/en/content/learn/modules/business_process_automation (accédé le 15 février 2020)

[6] Trailhead, module sur les bases de données.
https://trailhead.Salesforce.com/en/content/learn/modules/data_modeling/object_relationships (accédé le 15 février 2020)

[7] Appvizer, théorie des CRM.
https://www.appvizer.fr/magazine/relation-client/customer-relationship-management-crm/qu-est-ce-qu-un-crm#qu-est-ce-qu-un-crm-definition/ (accédé le 13 janvier mai 2020)

[8] Sage, caractéristiques des CRM.
https://www.sage.com/fr-fr/blog/choisir-logiciel-crm-erp/ (accédé le 20 janvier 2020)

[9] Versaccounts, l'histoire des ERP.
http://www.versaccounts.com/blog/the-history-of-erp-systems/ (accédé le 19 janvier 2020)

[10] Choisirmonerp, théorie des ERP.
https://www.choisirmonerp.com/erp/definition-d-un-erp (accédé 19 janvier 2020)

[11] Gestisoft, fonctionnalités des ERP.
https://gestisoft.com/les-fonctionnalites-dun-logiciel-erp-un-resume-complet/ (accédé le 20 janvier 2020)

[12] Rishabhsoft, les bases du cloud computing.
https://www.rishabhsoft.com/ (accédé le 2 février 2020)

[13] Packthub, l'architecture du cloud.
https://subscription.packtpub.com/ (accédé le 3 février 2020)

[14] Lebigdata, la définition du cloud.
https://www.lebigdata.fr/definition-cloud-computing
(accédé le 7 février 2020)

[15] Culture-informatique, les caractéristiques du cloud.
https://www.culture-informatique.net/cest-quoi-le-cloud/
(accédé le 7 février 2020)

[16] Macmillan education bookstore, les services du cloud.
https://www.macmillaneducationbookstore.com/overview-to-cloud-computing-services/ (accédé le 15 mai 2020)

[17] Weka, un modèle de contrat de bail.
https://www.weka.ch/out/pictures/master/product/1/contrats-de-bail-commercial.jpg (accédé le 15 mai 2020)