Un système d'automatisation de maison connectée, également appelé système domotique, est un ensemble de dispositifs électroniques, de capteurs, de logiciels et de technologies de communication conçus pour automatiser et simplifier la gestion et le contrôle des fonctions de la maison. L'objectif principal est d'améliorer le confort, la sécurité, l'efficacité énergétique et la commodité des habitations en permettant aux propriétaires de contrôler divers aspects de leur maison à distance, souvent via des applications mobiles ou des commandes vocales.

Comment va s'organiser l'élaboration du document de spécification des exigences (cahier des charges) :

Avant de commencer la rédaction du cahier des charges, il faut l'organiser. Cela inclut la définition des étapes, la création d'un calendrier de rédaction, et l'attribution des responsabilités aux membres de l'équipe ou aux parties prenantes impliquées dans le projet. Dans un premier temps, nous allons collecter toutes les informations dont nous avons besoin. Cela implique des réunions avec le client et les parties prenantes pour comprendre leurs besoins, objectifs, et attentes par rapport au projet. Une fois les informations collectées, nous devons analyser les besoins pour identifier les exigences spécifiques du projet. Cela inclut la différenciation entre les exigences métier, les exigences fonctionnelles, et les exigences non fonctionnelles.

Nous passons ensuite à la rédaction du document. Les différentes sections se présentent comme suit :

- Une introduction qui explique le but du document, son contexte et sa portée.

- Une section qui détaille les objectifs du projet et les buts à atteindre. Il s'agit de préciser ce que le projet vise à réaliser et quelles sont les attentes du client.

- Le cœur du document, une section qui répertorie toutes les exigences du projet, en les classant en exigences métier, exigences fonctionnelles et exigences non fonctionnelles. Chaque exigence doit être numérotée, suivie d'une description détaillée, et éventuellement accompagnée de cas d'utilisation, de scénarios et de critères de validation.

- Une section qui comporte des éléments d'architecture technique ou des composants clés, elle détaille comment ces éléments s'intègrent et interagissent pour répondre aux exigences du projet. Elle peut inclure des schémas, des diagrammes, ou des descriptions techniques.

- Une section qui énumère toutes les contraintes qui doivent être prises en compte lors du développement du projet. Cela peut inclure des contraintes techniques, budgétaires, juridiques, ou de calendrier.

- Une section qui spécifie les rôles et les responsabilités des différentes parties impliquées dans le projet. Cette section détaille qui est en charge de quoi et quelles sont les interactions entre les parties.

-Si le calendrier du projet n'est pas détaillé dans une section distincte, il peut être mentionné ici, en précisant les échéances clés.

Une fois le premier brouillon du cahier des charges rédigé, nous allons le soumettre à des revues et révisions par des pairs ou des experts pour s'assurer de sa qualité, de sa cohérence, et de son exhaustivité. Puis, le cahier des charges doit être soumis au client ou à l'utilisateur final pour validation. Le client doit approuver le document pour confirmer que les besoins et attentes sont correctement reflétés.

Le plan de conception architecturale (Dossier d'architecture technique) :

Afin de réussir la conception de l'architecture technique pour notre projet, nous allons suivre un processus structuré composé de plusieurs étapes clés. Ces étapes ont pour objectif de créer une conception architecturale solide qui réponde aux besoins et aux attentes du client. Notre premier pas consiste à effectuer une analyse approfondie des besoins du projet à partir du cahier des charges. Cela comprend la compréhension des exigences métier, fonctionnelles et non fonctionnelles du logiciel. Nous allons identifier les fonctionnalités requises, les contraintes, les objectifs, et les attentes du client. Une fois les besoins identifiés, nous mènerons une étude de faisabilité pour évaluer la viabilité du projet. Cette étape nous permettra de déterminer si les exigences peuvent être satisfaites compte tenu des contraintes de temps, de budget et de technologie. Après avoir évalué la faisabilité, nous sélectionnerons les technologies, les langages de programmation, les frameworks, les bases de données, et d'autres outils qui seront utilisés pour le développement du logiciel. Le choix des technologies devra être en adéquation avec les besoins du projet. Une fois les technologies choisies, nous allons entamer la conception de l'architecture globale du logiciel. Cette étape implique de déterminer comment les différents composants interagiront et se connecteront pour répondre aux exigences du projet. Des diagrammes d'architecture peuvent être élaborés pour représenter cette structure.
Dans un dernier temps, nous allons élaborer une conception préliminaire du logiciel. Bien que les grandes lignes de l'architecture soient définies, les détails techniques ne sont pas encore spécifiés. Cette phase vise à créer une vision globale du système afin de commencer sereinement la partie code.

Planification de l'implémentation du code et les tests associés :

Une fois que la conception est validée, notre équipe passera à la mise en œuvre technique. Cette phase est souvent la plus courte du cycle en V, car elle consiste à traduire nos plans en code concret. Les développeurs joueront un rôle essentiel à ce stade. Le "Code Source" constitue le cœur de notre logiciel, mettant en œuvre de manière concrète les spécifications techniques que nous avons établies lors de la phase de conception. Il est essentiel que nos développeurs rédigent un code de haute qualité. De plus, nous devons intégrer dans le code des tests associés pour vérifier son bon fonctionnement. Les tests sont essentiels pour s'assurer que le code répond aux exigences et pour détecter rapidement d'éventuelles erreurs. En plus du code source et des tests, la "Documentation Tech" revêt une importance cruciale. Elle va au-delà du code en lui-même. Elle expliquera le fonctionnement des fonctions, des classes, des méthodes, des variables, et fournira des commentaires utiles pour que d'autres membres de l'équipe puissent comprendre et collaborer efficacement sur le code. La documentation technique jouera un rôle clé dans notre travail d'équipe.

Préparation des tests unitaires, d'intégration et fonctionnels :

Pour les tests unitaires, il faut dans un premier temps identifier la plus petite unité de code à tester, généralement une fonction ou une méthode. Assurez-vous que l'unité est suffisamment petite pour être testée de manière isolée. Avant de commencer à écrire des tests, il est crucial de comprendre les exigences fonctionnelles de l'unité que vous testez. Quels sont les comportements attendus, les entrées acceptables, et les résultats prévus ?
Créez un environnement de test isolé où vous pourrez exécuter les tests de manière cohérente. Cela peut inclure la configuration de bases de données de test, de fichiers temporaires, ou de tout autre élément nécessaire. Puis, rédigez les cas de test pour l'unité. Un bon cas de test doit couvrir différents scénarios, y compris les cas normaux, les cas limites et les cas d'erreur. Préparez les données de test spécifiques à l'unité que vous testez et assurez vous qu'elles soient reproductibles en production. Lancez les tests de manière isolée et analysez les résultats.

Pour les tests d'intégration, il faut déterminer les différents composants ou modules de votre application qui nécessitent des tests d'intégration. Ces composants peuvent inclure des services, des API, des bases de données, des interfaces utilisateur, ou toute autre partie critique de l'application. Il est essentiel de comprendre comment ces composants interagissent les uns avec les autres. Identifiez les dépendances, les flux de données, les API utilisées, et les scénarios d'utilisation typiques. Créez des cas de test spécifiques pour chaque interaction clé entre les composants.

Les cas de test devraient couvrir différents scénarios, y compris les cas normaux, les cas limites, et les cas d'erreur. Il faut que l'environnement de test reflète l'environnement de production autant que possible. Bien sûr, assurez-vous d'avoir des ensembles de données de test pertinents pour chaque interaction. Vous pouvez ensuite lancer les tests d'intégration en utilisant l'environnement de test et les données de test que vous avez préparés et pour finir, analyser les résultats.

Pour les tests fonctionnels, il faut déterminer les fonctionnalités de l'application ou du système qui doivent être testées. Cela peut inclure des fonctionnalités spécifiques, des cas d'utilisation et des scénarios d'utilisation. Avant de commencer à rédiger des tests, assurez-vous de comprendre en détail les exigences fonctionnelles de chaque fonctionnalité que vous testez. Quels sont les comportements attendus, les entrées acceptables, et les résultats prévus ? Rédigez des cas de test spécifiques pour chaque fonctionnalité. Ces cas de test doivent définir les étapes à suivre, les données d'entrée à utiliser, et les résultats attendus pour chaque scénario. Il faut choisir des données de test pertinentes pour chaque cas de test et s'assurer que ces données reflètent des situations réalistes et variées. L'environnement de test doit refléter au maximum l'environnement de production. Pour finir, il faut exécuter puis analyser les tests.

Planification de la maintenance et du support du système une fois déployé :

On peut considérer le déploiement comme l'étape à la suite du processus de développement et la maintenance comme son étape test associée. Pour mettre en route la maintenance, les utilisateurs finaux et les membres de l'équipe signalent les problèmes, les anomalies ou les besoins d'évolution. Puis, les problèmes signalés sont évalués pour déterminer leur gravité et leur impact sur le système. L'équipe de développement développe des correctifs, des mises à jour ou de nouvelles fonctionnalités en fonction des problèmes identifiés et des besoins d'évolution. S'ensuit, une phase de test où les correctifs ou les mises à jour sont soumis à des tests approfondis pour s'assurer qu'ils ne causent pas de nouveaux problèmes et qu'ils répondent aux besoins. Pour finir, les correctifs ou les mises à jour sont déployés dans l'environnement de production. Toutes les modifications et les procédures sont soigneusement documentées pour assurer la traçabilité. Pour terminer la boucle, l'équipe de maintenance fournit une assistance aux utilisateurs finaux en faisant remonter les problèmes afin qu'ils puissent être résolus par le même processus.

Sources :

https://asana.com/fr/resources/software-requirement-document-templatehttps://www.proinfluent.com/cycle-en-v/
https://www.atlassian.com/fr/continuous-delivery/software-testing/types-of-software-testing
https://www.ionos.fr/digitalguide/sites-internet/developpement-web/le-cycle-en-v/
https://hal.science/cel-02004689/document
https://www.techno-science.net/definition/711.html
https://blog.hubspot.fr/marketing/cahier-des-charges-fonctionnels
https://chef-de-projet.fr/specification-fonctionnelle-et-cahier-des-charges/
https://web.archive.org/web/20160705043334/http://www.in2p3.fr/actions/formation/ConduiteProjet06/doc-dialinas.pdf