

Projet Galt

Spécifications

Version .2

18/10/2016

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 13/10/2016 | 0.1 | Première description | AC |
| 18/10/2016 | 0.2 | Cas d’utilisation réalisés | AC |
|  |  |  |  |
|  |  |  |  |

Sommaire

1 Introduction 6

1.1 Contexte initiale, historique et vision 6

1.2 Mission 6

1.3 Objectifs 6

1.4 Glossaire 6

1.5 Documents de référence 6

2 Description générale 7

2.1 Acteurs 7

2.2 Cas d’utilisations 7

3 Spécifications fonctionnelles 9

3.1 Carte de navigation 9

3.2 Détails des cas d’utilisations 9

4 Spécifications non fonctionnelles 13

4.1 Environnement opérationnel 13

4.2 Contraintes de conception et d’implémentation 13

4.3 Documentation utilisateur 13

4.4 Exigences de performance 14

4.5 Exigences de sécurité 14

4.6 Attributs de qualité 14

Les spécifications sont à la base de l’accord entre le client et l’équipe réalisatrice sur ce que le système devra faire et ce qu’il ne devra pas faire.

Ce document est adapté à une description des fonctionnalités sous forme de cas d’utilisation.

Critères d’évaluations lors de l’avant-projet ou avant chaque début d’itération (\*) :

Spécifications fonctionnelles :

* Les acteurs (visiteur, administrateur, …) sont identifiés
* Les cas d’utilisations sont listés
* Les cas d’utilisations ont été détaillés + maquette (\*\*)
* La carte de navigation du logiciel est présente (\*)

Spécifications non-fonctionnelles

* Les contraintes de conception et d’implémentation ont été listées
* Les attributs de qualité ont été listés

(\*) Dans le cas d’un processus itératif, les spécifications détaillées (paragraphe 3) peuvent concerner l’itération courante ou plusieurs itérations

(\*\*) Dans le cas d’un processus itératif, les spécifications détaillées sont validés par le client à chaque début d’itération

# Introduction

## Contexte initial, historique et vision

Lors du stage de Thibaud, une responsabilité dans un projet lui a été attribuée. Le projet avait un réel intérêt mais malheureusement avait un développement assez lent et n’a toujours pas abouti. La motivation étant de recommencer le projet avec une équipe complète et de le réaliser en repartant de zéro avec des technologies mieux adaptées.

## Mission

Création d’un site web permettant d’afficher le graphe des dépendances d’un package.

## Objectifs

1. Elaboration d’une intégration continue à l’aide d’une chaine de build dès le commencement l’itération 1.

2. Création d’un serveur web .NET en modèle MVC gérant le client et la base de données d’ici la fin de l’itération 1.

3. Création d’une base de données azure permettant d’archiver les dépendances d’un package et l’administrer tout au long du projet.

4. Création d’un client web avec le Framework JavaScript Vuejs d’ici la fin de l’itération 2.

5. Afficher les graphes des dépendances à l’aide de la technologie web D3js d’ici la fin de l’itération 3.

## Glossaire

Définir les termes propres au domaine de l'application, les sigles et les abréviations, les termes techniques nécessaires à une bonne compréhension et interprétation du document

### Termes du domaine

Dépendance : Intégration d’un package *x* au sein d’un autre package *y*. On dit que le package *y* dépend de *x*.

Package : Archive contenant des fichiers et des informations nécessaires à l’installation d’un logiciel.

NuGet : Gestionnaire de packages en ligne. Il sera celui utilisé sur le site Galt.

Graphe : Ensemble de points reliés par des flèches. Nous utiliserons ici des graphes de dépendances qui représentent l’architecture des dépendances d’un package.

### Termes techniques

Chaîne de build : Pratique permettant de vérifier à chaque modification du code source que les modifications ne créent pas une régression dans le logiciel.

C# : Langage qui permet de réaliser des sites web dynamiques.

.NET : framework facilitant la conception d’applications web.

JS (Javascript) : Langage utilisé sur les pages web, s’exécutant côté client.

Vue.js : framework facilitant la conception de SPA.

D3.js : bibliothèque Javascript permettant l’affichage de données graphiques et dynamiques.

TypeScript : Langage simplifiant la création d’applications web. Il s’agit d’un sur-ensemble de JS

Azure : Service cloud permettant le stockage de données

SPA (Single Page Application) : application web accessible via une page web unique

Modèle MVC (Model, View, Controller) : Manière d’organiser son code de façon à respecter le principe de séparation des responsabilités dans une application.

Côté client / côté serveur : Le côté client correspond aux langages s’exécutant sur la machine de l’Utilisateur ; le côté serveur correspond aux langages s’exécutant sur le Système.

# Description générale

## Acteurs

Présenter les différentes familles d’utilisateurs de la solution

|  |  |
| --- | --- |
| **Nom** | **Description** |
| Utilisateurs | Développeurs ayant besoin de connaître l’ensemble des dépendances de leur package ainsi que l’actualité des mises à jour de celles-ci |

## Cas d’utilisations

Lister les cas d’utilisations principales du produit, les détails seront donnés plus bas

### Diagramme de cas d’utilisation

Insérer ici le diagramme de cas d’utilisation représentant les cas d’utilisation (détaillés dans le 3) associé aux acteurs. Structurer éventuellement le diagramme en utilisant des relations d’inclusion, d’extension ou de généralisation.

### Listes des cas d’utilisations

Nommer et décrire brièvement les cas d’utilisation répondant aux exigences fonctionnelles de votre système.

Pour identifier les cas d'utilisation appropriés, commencez par réfléchir à ce que chaque acteur attend du système. Pour chaque acteur, humain ou non, posez-vous les questions suivantes :

* Du point de vue de l'acteur, quelles sont les principales tâches que le système doit exécuter ?
* L'acteur va-t-il créer, stocker, modifier, supprimer ou lire des données dans le système ?
* L'acteur sera-t-il amené à informer le système de changements extérieurs soudains ?
* L'acteur doit-il être avisé de l'occurrence de certains événements dans le système ?
* Une fois les cas d’utilisation identifiés, hiérarchisez-les en tenant compte des deux facteurs suivants :
* La priorité fonctionnelle
* Le risque technique
* L’objectif ensuite est de planifier en premier (premières itérations) la réalisation des cas d’utilisation ayant un risque élevé et une priorité haute.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom du cas d’utilisation** | **Acteur(s)** | **Priorité** | **Risques** |
| Arriver sur le site | Utilisateur  Système | Haute | Faible |
| Faire une recherche de package | Utilisateur  Système | Moyenne | Haut |
| Afficher tous les résultats d’une recherche | Utilisateur  Système | Moyenne | Faible |
| Afficher un package | Système | Haute | Haut |
| Changer de version du package | Utilisateur  Système | Faible | Faible |

# Spécifications fonctionnelles

## Carte de navigation

L'objectif de la carte de navigation est d'exprimer les chemins d'interface principaux dans le système. Ces chemins sont les chemins principaux à l'écran et ne reprennent pas la totalité des possibilités. Elle peut être considérée comme une carte routière de l'interface utilisateur.

Un diagramme de navigation peut être représenté grâce à un diagramme d’activité

Nous avons réalisé notre diagramme de navigation sur le site Lucidchart :

<https://www.lucidchart.com/invitations/accept/1e09256e-a96f-477b-a76a-8832e0678b13>

Nous avons également réalisé des maquettes de nos vues :

<https://classic.moqups.com/acharrier@intechinfo.fr/W1B9DKyP>

## Détails des cas d’utilisations

Certains cas d’utilisation complexe avec de nombreuses extensions et/ou répétitions peuvent être illustrés par un diagramme d’activité.

### Nom du cas

Le nom du cas doit commencer par un verbe. Il correspond à l’objectif à atteindre.

Arriver sur le site

#### Description :

Résumé succinct qui présente le scénario de base (succès) dans un paragraphe.

Peut être rédigé lors de la première étude de besoins et avant la rédaction détaillée du cas d’utilisation pour se faire une idée du sujet et de son périmètre.

La page d’accueil s’affiche

#### Acteur principal :

Celui qui fait appel au système pour atteindre un but.

Un acteur n’est pas un utilisateur : une même personne peut jouer plusieurs rôles, plusieurs personnes peuvent jouer un même rôle.

Un acteur peut être un système, une machine.

Le serveur qui renvoie la page à l’utilisateur.

#### Acteurs secondaires :

Optionnel.

Autres participants du cas d’utilisation.

L’utilisateur qui a envoyé la requête.

#### Préconditions :

Ce qui doit toujours être vrai avant le début d’un scénario. En principe, une précondition implique que le scénario d’un autre cas d’utilisation s’est déroulé normalement (ex : « Le Caissier est identifié et authentifié »).

#### Postconditions (garantie en cas de succès) :

Ce qui doit être vrai lorsque le cas d’utilisation se termine avec succès, qu’il s’agisse du scénario principal ou d’un scénario alternatif.

La page d’accueil s’est correctement affichée

#### Scénario nominal :

Il décrit le scénario type qui satisfait les intérêts de l’acteur principal et des acteurs secondaires. Il ne comprend pas de conditions ni de branchement. Les traitements conditionnels sont à reporter dans la section Extensions.

1. Le cas débute lorsque …

La première étape indique l’événement qui déclenche le scénario.

Exemple pour le cas d’utilisation Traiter une vente : « Le Client arrive à la caisse avec les articles qu’il souhaite acheter »

l’utilisateur tape l’URL dans sa putain de barre d’adresse

1. L’Utilisateur {verbe d’action} … tape l’URL du site et envoie une requête au système
2. Le Système {verbe d’action} … lui renvoie la page
3. L’Utilisateur reçoit la page envoyée par le système

Le nom des acteurs doivent commencer par une majuscule.

Chaque étape d’actions est rédigée comme une action simple « dans laquelle le sujet est actif ». On peut la comparer à la description d’un match de football : « la pêrsonne1 envoie le ballon à la personne2 ; la personne 2 dribble ; la personne2 envoie le ballon à la personne3 ».

Le scénario est composé d’étapes, lesquelles sont de trois sortes :

1. Une interaction entre deux acteurs (le système étant considéré comme un acteur). Exemple : « Le client saisit son adresse », « Le Système affiche le prix de l’article »

2. Une étape de validation pour protéger les intérêts d’un acteur. Exemple : « Le Système valide le code secret ».

3. Un changement interne pour satisfaire aux intérêts d’un intervenant. Exemple : « Le Système enregistre la vente ».

Il peut être parfois utile d’utiliser des mots clés permettant d’exprimer la répétition

Exemple :

Pour chaque article trouvé

Le système affiche une image du produit

Le système affiche le prix du produit

…

1. Pour {expression d’itération}
2. Faire quelque chose
3. Faire quelque chose d’autre
4. Tant que {expression booléenne}
5. Faire quelque chose
6. Faire quelque chose d’autre

#### Extensions:

Les extensions permettent de décrire les autres scénarios ou branchement possibles, tant en cas de succès qu’en cas d’échec.

3a . Si {*Condition d’extension de l’étape 3 du scénario nominal– à compéter}*  alors:

Comme pour la première étape du scénario nominal, la condition d’extension indique l’événement qui déclenche le scénario.

Exemple : « Si le code de l’article est invalide (non trouvé par le Système) alors »



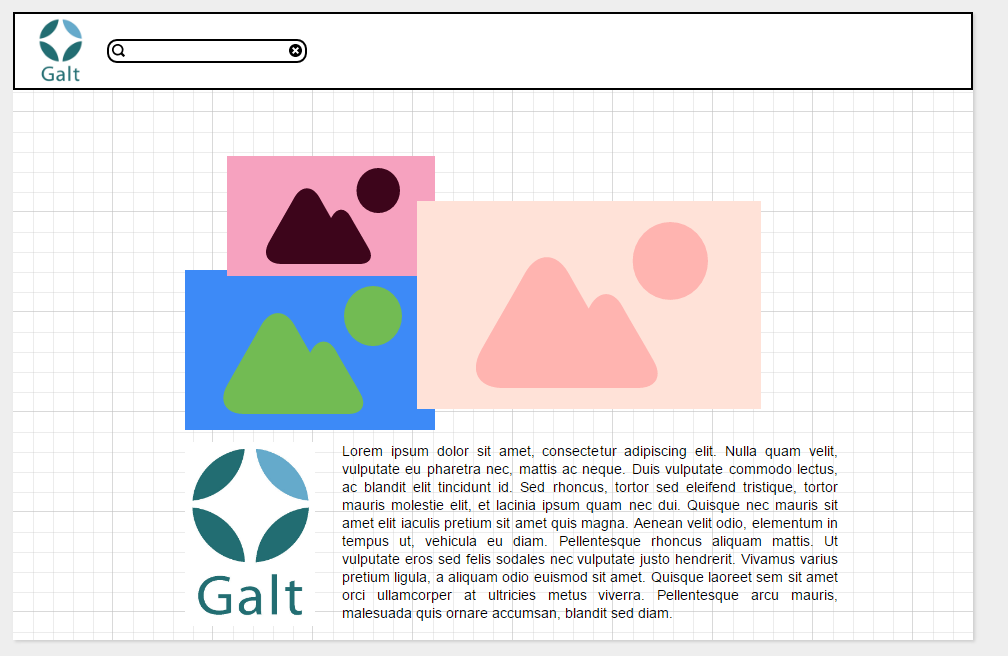
Description des étapes permettant de prendre en charge la condition d’extension

#### Interface utilisateur:

Représenter les écrans de l’application. Il convient de rappeler que la maquette n’est pas le produit final !

Vous pouvez pour cela dessiner les écrans à la main ou utiliser un outil spécifique de création d’interface utilisateur (exemple : <http://www.mockupscreens.com/>, http://c2.com/cgi/wiki?GuiPrototypingTools)

Page d’accueil



#### Fréquence :

Optionnel.

Fréquence moyenne estimé d’apparition de ce cas d’utilisation par jours

#### Spécifications particulières :

Optionnel.

Besoin non fonctionnel, attribut de qualité ou contrainte se rapportant spécifiquement à ce cas d’utilisation. Il peut s’agir de questions de performance, de fiabilité ou d’ergonomie. Exemple : « Interface utilisateur à écran tactile sur grand écran plat. Le texte doit être visible à un mètre. ».

#### Questions en suspens:

Optionnel.

### Nom du cas

Faire une recherche de package

#### Description :

L’utilisateur tape des mots-clefs dans la barre de recherche. Le système lui renvoie des propositions de package.

#### Acteur principal :

L’utilisateur qui effectue la recherche.

#### Acteurs secondaires :

Le système qui renvoie les propositions de package.

#### Préconditions :

La page d’accueil s’est affichée (ou n’importe quelle autre page car la recherche est disponible depuis chaque page du site)

#### Postconditions (garantie en cas de succès) :

Le système a soit affiché la page d’un package, soit affiché la page « More results… »

#### Scénario nominal :

1. Le cas débute lorsque …l’utilisateur clique dans la barre de recherche
2. L’Utilisateur {verbe d’action} … tape des mots-clefs
3. Le Système {verbe d’action} … affiche sous la barre de recherche les différents résultats en rapport avec les mots-clefs ainsi qu’une option « More results… »
4. L’utilisateur clique sur un des résultats ou modifie sa recherche
5. Tant que l’utilisateur ne tape pas de mots-clefs
6. Le Système n’affiche rien

#### Extensions:

3a . Si l’utilisateur clique sur un des résultats alors:

1. Le Système enlève les résultats
2. Le Système affiche la page du package correspondant au résultat choisi par l’Utilisateur

3b . Si l’utilisateur clique sur « More results… » alors :

1. Le Système enlève les résultats
2. Le Système affiche la page de tous les résultats de la recherche

3c . Si l’utilisateur modifie sa recherche alors :

1. Le Système affiche les nouveaux résultats de la recherche sous la barre de recherche ainsi que « More results… »

3d . Si le Système ne trouve aucun résultat lors de la recherche alors :

1. Le Système affiche seulement « More results… »
2. Si l’Utilisateur clique sur « More results… », le Système affiche une page de résultats vide avec un message expliquant qu’aucun résultat n’a été trouvé

3e . Si l’utilisateur clique sur « More results…» alors :

1. Le Système enlève les résultats
2. Le Système affiche la page de tous les résultats de la recherche\*

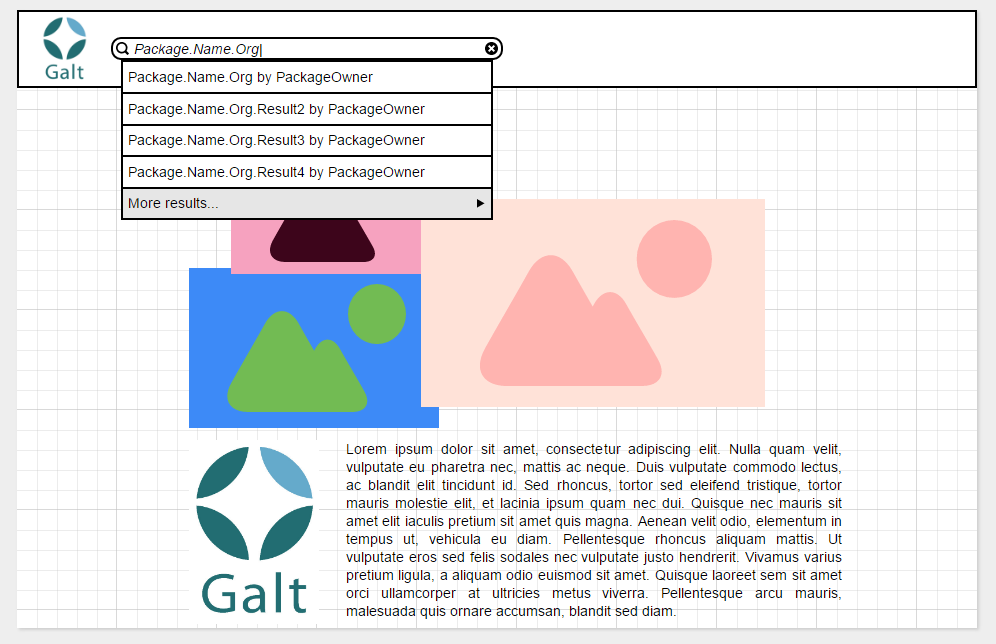
3f . Si l’utilisateur supprime les mots-clefs dans la barre de recherche alors :

1. Le Système enlève le menu des résultats de la recherche

#### Interface utilisateur:

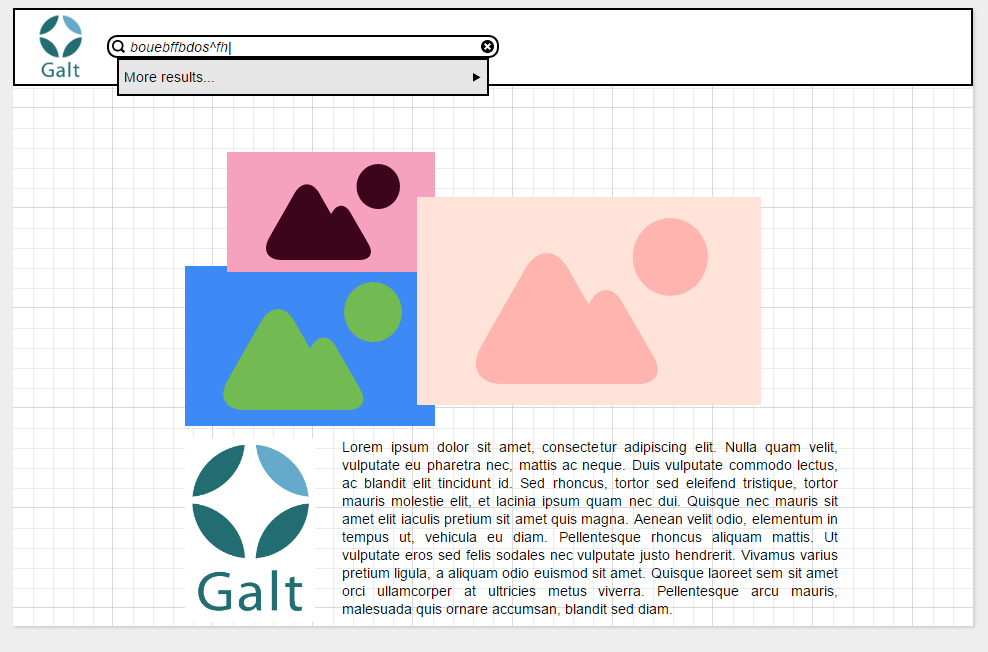
Représenter les écrans de l’application. Il convient de rappeler que la maquette n’est pas le produit final !

Vous pouvez pour cela dessiner les écrans à la main ou utiliser un outil spécifique de création d’interface utilisateur (exemple : <http://www.mockupscreens.com/>, http://c2.com/cgi/wiki?GuiPrototypingTools)



Page d’accueil lors d’une recherche

Page d’accueil lors d’une recherche sans résultats



#### Fréquence :

Régulière

### Nom du cas

Afficher tous les résultats d’une recherche

#### Description :

L’utilisateur cherche plus de résultats lors de sa recherche

#### Acteur principal :

L’utilisateur qui clique sur « More results… ».

#### Acteurs secondaires :

Le système qui renvoie la page des résultats

#### Préconditions :

L’utilisateur a effectué une recherche et a cliqué sur « More results… »

#### Postconditions (garantie en cas de succès) :

Le système a affiché la page du package choisi par l’utilisateur

#### Scénario nominal :

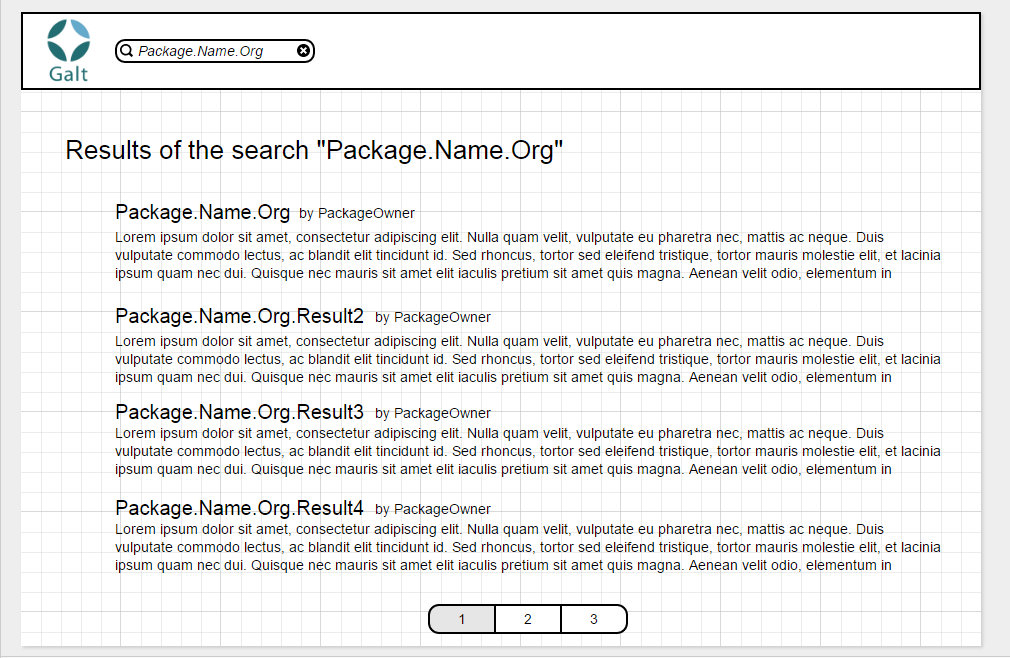
1. Le cas débute lorsque l’Utilisateur clique sur « More results… » après avoir effectué une recherche
2. Le Système {verbe d’action} … renvoie la page des résultats de la recherche
3. L’Utilisateur clique sur le package désiré

#### Extensions:

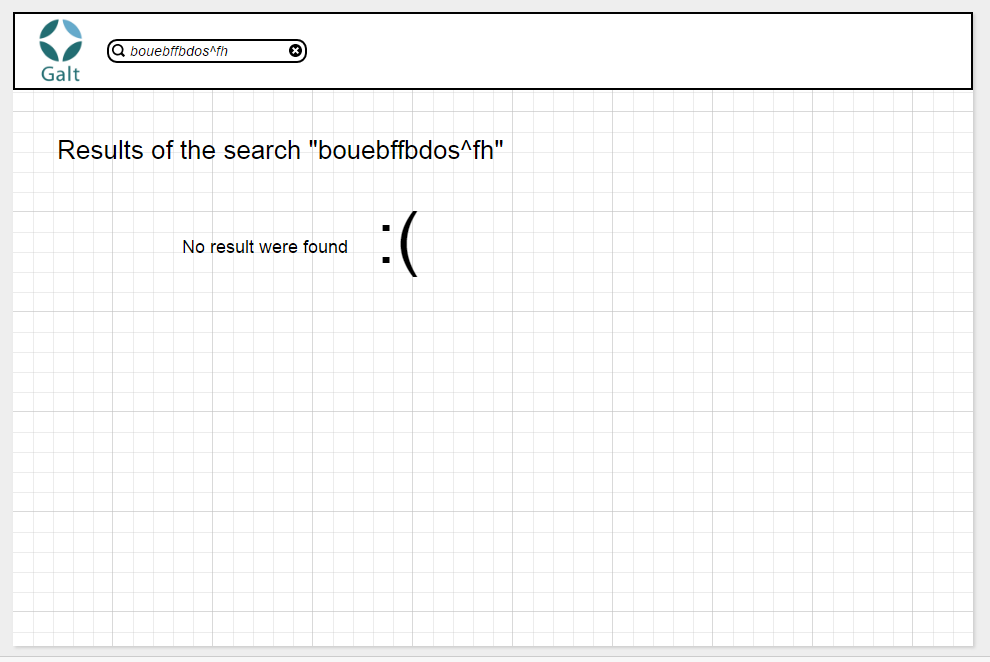
3a . Si l’utilisateur a effectué une recherche qui conduit à aucun résultat :

1. Le Système renvoie la page des résultats avec un message indiquant qu’aucun résultat n’a été trouvé

#### Interface utilisateur:



Page des résultats d’une recherche



Page des résultats d’une recherche sans aucun résultat

#### Fréquence :

Moyenne

### Nom du cas

Afficher un package

#### Description :

L’utilisateur a effectué une recherche et a cliqué sur un résultat

#### Acteur principal :

Le Système qui affiche le package

#### Préconditions :

L’utilisateur a cliqué sur un package

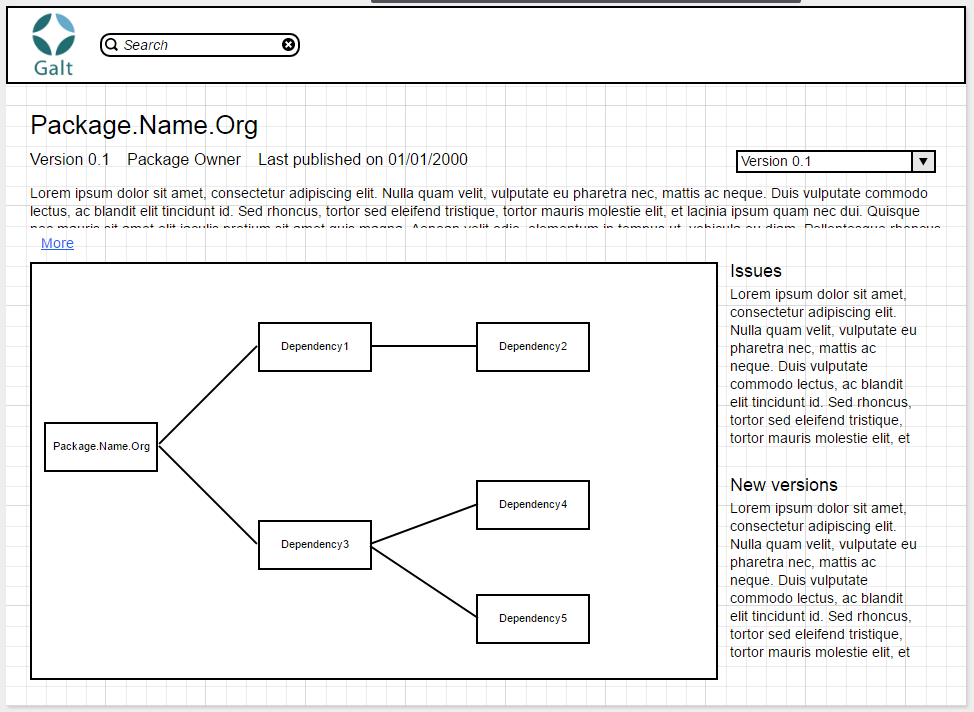
#### Postconditions (garantie en cas de succès) :

Le système a affiché la page du package choisi par l’utilisateur

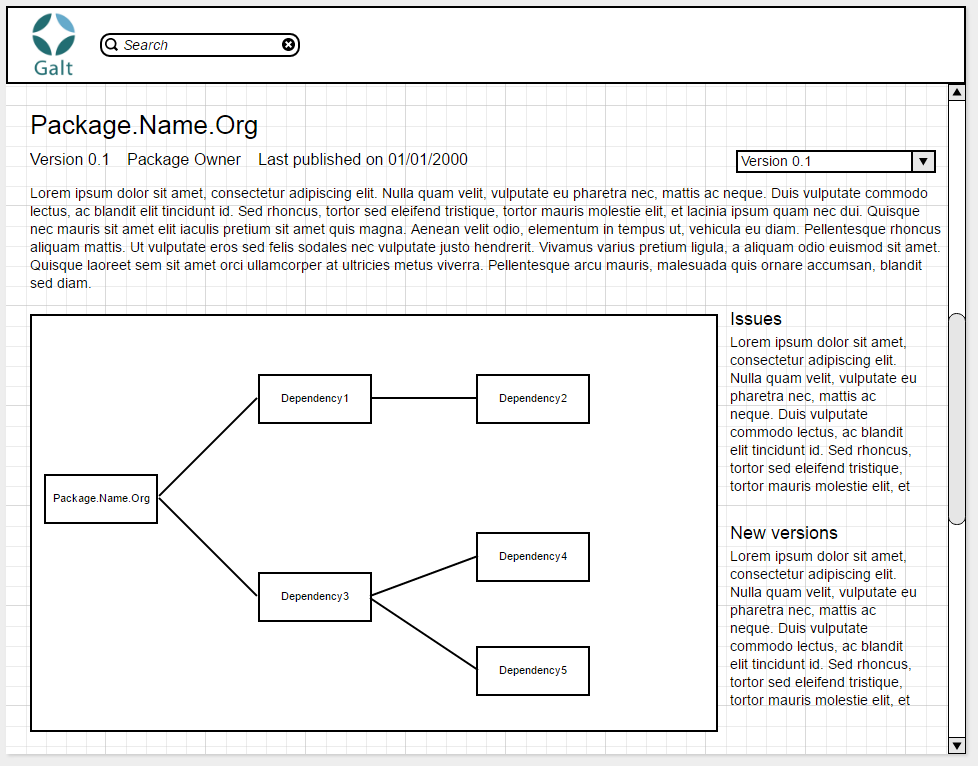
#### Scénario nominal :

1. Le cas débute lorsque l’Utilisateur clique sur un package dans une recherche
2. Le Système {verbe d’action} … renvoie la page du package

#### Interface utilisateur:



Page de package



Page du package après avoir cliqué sur « More » (description étendue)

#### Fréquence :

Fréquente

### Nom du cas

Changer de version du package

#### Description :

L’utilisateur souhaite changer la version du package

#### Acteur principal :

L’Utilisateur qui change la version du package

#### Acteurs secondaires :

Le Système qui affiche la nouvelle version du package

#### Préconditions :

L’Utilisateur est sur la page d’un package en version *x*

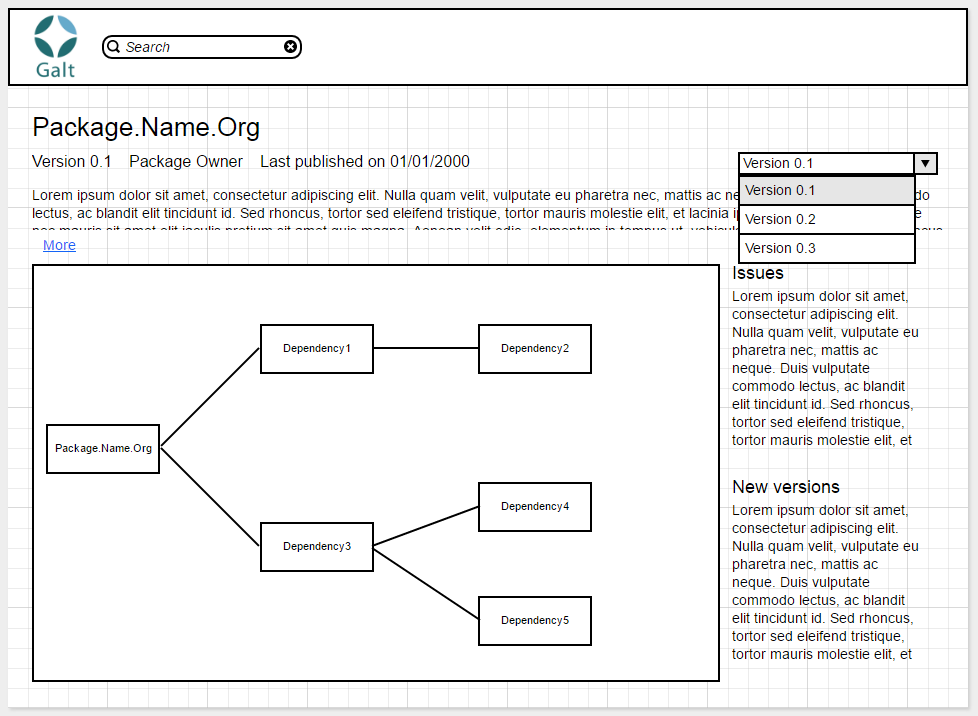
#### Postconditions (garantie en cas de succès) :

L’Utilisateur est sur la page du même package en version *y*

#### Scénario nominal :

1. Le cas débute lorsque l’Utilisateur clique sur le menu déroulant des versions
2. L’Utilisateur clique sur la version souhaitée dans le menu déroulant
3. Le Système affiche la nouvelle page avec la version demandée par l’Utilisateur

#### Interface utilisateur:



Page de package lors d’un changement de version

#### Fréquence :

Fréquente

# Spécifications non fonctionnelles

## Environnement opérationnel

Décrire l’environnement dans lequel le produit devra fonctionner.

Pour une application informatique par exemple : le matériel, le système d’exploitation et sa version, la localisation géographique des utilisateurs, les serveurs, les bases de données, etc.

Lister les autres produits, systèmes, applications informatiques, avec lesquels le produit devra coexister pacifiquement

L’équipe utilisera le logiciel Microsoft Visual Studio comme environnement de travail et Windows comme système d’exploitation (versions 7, 8.1 et 10). Le site sera accessible pour des utilisateurs du monde entier. Nous utiliserons le service Microsoft Azure pour notre base de données. L’application devra coexister pacifiquement avec NuGet.

## Contraintes de conception et d’implémentation

Décrire tous les facteurs qui vont restreindre la liberté des concepteurs, donner la raison de chaque contrainte

Types de contraintes :

* Technologies, outils, langages de programmation, bases de données, spécifiques imposées ou interdites
* Restriction sur la version du système d’exploitation ou du navigateur Web
* Règles ou conventions imposées dans l’entreprise : structure de la documentation technique destinée à une future maintenance par exemple
* Compatibilité avec des produits plus anciens
* Limitations imposées par les règles du métier
* Limitations liées au matériel disponible : temps de réponse, taille mémoire, vitesse du processeur, taille, poids, matériaux, coûts
* Conventions liées à l’interface utilisateur, lors d’une extension par exemple
* Format standards d’échange d’information : XML par exemple

## Documentation utilisateur

Lister les composants de la documentation à destination des utilisateurs qui sera livrée avec le produit

Ceci comprend : les manuels utilisateurs, l’aide en ligne et les tutoriaux

Identifier pour chaque document le format, les standards à respecter, les outils à utiliser

## Attributs de qualité

Indiquer ici tous les autres attributs de qualité importants pour l’utilisateur et les développeurs. Ces caractéristiques doivent être spécifiques, quantitatives et vérifiables

Indiquer les priorités des divers attributs. Par exemple : « la facilité d’apprentissage est plus importante que la facilité d’installation »

Les attributs de qualité importants pour les utilisateurs sont :

* Performances spécifiques nécessaires à la bonne marche du produit. Donner les raisons pour faciliter le travail des concepteurs. Quantifier au maximum
* Disponibilité : up-time
* Efficience : bonne utilisation des ressources
* Flexibilité : extension, augmentation, réduction, etc.
* Intégrité : gestion des accès, protection des données, etc.
* Interopérabilité : échange de données et de services avec d’autres systèmes
* Fiabilité : probabilité d’exécution sans erreur
* Robustesse : tolérance aux fautes (Tankness en argot de développeur US)
* Utilisabilité : facilité d’utilisation (user friendly)
* Sécurité : toute exigence concernant la sécurité, l’intégrité, le respect de la vie privée. Considérer le produit, l’usage du produit, les données que le produit utilise ou crée.

Les attributs de qualité importants pour les développeurs sont :

* Maintenabilité : facilité de correction ou de modification
* Portabilité : facilité de migration d’un environnement à un autre
* Ré-utilisabilité : notion de composant
* Testabilité : facilité de test.