

**BCEPHAL-PM-SDD-001**

**B-CEPHAL**



**Software Design Documents**

Moriset & co

www.moriset.com

Table des matières

[Table des tableaux ii](#_Toc155512498)

[Table des figures ii](#_Toc155512499)

[1. introduction 1](#_Toc155512500)

[1.1. But du document 1](#_Toc155512501)

[1.2. Cible 1](#_Toc155512502)

[1.3. Portée du projet 1](#_Toc155512503)

[1.4. Définitions 1](#_Toc155512504)

[1.5. Acronymes et abréviations 1](#_Toc155512505)

[1.6. Documents de référence 1](#_Toc155512506)

[2. Modele conceptuel 2](#_Toc155512507)

[2.1. contexte 2](#_Toc155512508)

[2.2. le cycle de vie 2](#_Toc155512509)

[2.2.1. Gestion des projets 2](#_Toc155512510)

[2.3. Les entités 2](#_Toc155512511)

[2.3.1. La classe ProjectVersion 2](#_Toc155512512)

[2.3.2. La classe Module 3](#_Toc155512513)

[2.3.3. La classe Functionality 3](#_Toc155512514)

[2.3.4. La classe Issue 4](#_Toc155512515)

[2.3.5. La classe TestScenario 4](#_Toc155512516)

[2.3.6. La classe TestCase 5](#_Toc155512517)

[2.3.7. La classe TestStep 5](#_Toc155512518)

[2.3.8. La classe TestCondition 5](#_Toc155512519)

[2.3.9. La classe TestData 6](#_Toc155512520)

[2.3.10. La classe TestExpectedResult 6](#_Toc155512521)

# Table des tableaux

[Tableau 1: Acronymes et abréviations 2](#_Toc45234097)

[Tableau 2: Document de référence 2](#_Toc45234098)

# Table des figures

Aucune entrée de table d'illustration n'a été trouvée.

# introduction

## But du document

Ce document fournit les définitions nécessaires pour conceptualiser et formaliser le design du module *Project Manager* de B-CEPHAL dont les besoins et ls fonctionnalités sont résumés dans le document de spécification des besoins (SRS).

## Cible

Ce document s’adresse à l’équipe de développement de B-CEPHAL.

## Portée du projet

Ce SDD contient la définition générale et les caractéristiques du projet, l'architecture globale du système et l'architecture des données, un calendrier d’avancement. A l'aide de diagrammes UML, la conception du module sera expliquée visuellement afin d'aider les développeurs à comprendre correctement et facilement toutes les informations indiquées dans ce document.

## Définitions

## Acronymes et abréviations

|  |  |
| --- | --- |
| Acronyme/Abréviation | Définition |
|  |  |
|  |  |
| SDD | Software Design Document |
| SRS | Software Requiments Specification |

Tableau 1: Acronymes et abréviations

## Documents de référence

|  |  |
| --- | --- |
| Référence | Titre |
|  |  |
|  |  |
|  |  |

Tableau 2: Document de référence

# Modele conceptuel

Cette section traite des termes de base, des concepts et du contexte du SDD.

## contexte

Le but de ce module est de fournir un environnement de gestion de projets et des facilités de communication entre les membres de l’équipe projet. En effet, il s’agit de gérer le cycle vie complet d’un projet de l’émission du besoin à la livraison en passant par la réalisation, les tests, la gestion des bugs et les évolutions.

Ce projet sera implémenté en C# (Blazor) pour ce qui est du client et en Java pour le serveur.

## le cycle de vie

### Gestion des projets

## Les entités

### La classe ProjectVersion

Cette classe représente une version d’une application.

#### Les champs

* **id** : LONG unique à chaque version. Cet id est généré par le système.
* **code** : STRING représentant la version du projet(exemple : V8.2.3). Il ne peut exister deux versions d’un même projet possédant le même code. Ce champ est obligatoire et est défini par l’utilisateur à la création. Sa taille est comprise entre 1 et 20 caractères.
* **name**: STRING dont la valeur représente le nom de la version du projet (exemple : Azobé). Le nom n’est pas obligatoire et est définit par l’utilisateur. Sa taille ne peut excéder 100 caractères.
* **description**: STRING dont la valeur représente la description de la version. Ce champ peut être nul.
* **group** : de type BGROUP, il représente le groupe
* **active**: Boolean indiquant si la version est actives ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Une version inactive est ignorée par le systèmes lors des différentes opérations.
* **deliveryDate**: La date de livraison. Elle peut être nulle et est définit pas l’utilisateur le cas échéant.
* **creationDate**:
* **modificationDate**:
* **phase**: La phase courante du cycle de vie de la version du projet. Il s’agit d’une énumération dont les valeurs possibles sont :
  + ANALYSE
  + IMPLEMENTATION
  + FACTORY\_TESTING
  + ACCEPTANCE\_TESTING
  + AWAITING\_GO\_NOGO
  + PRODUCTION
* **status**: Le statut de la version. Il s’agit d’une énumération dont les valeurs possibles sont :
  + OPENED
  + CLOSED
  + SUSPENDED
* **category**: La catégorie de la version. Il s’agit d’une énumération dont les valeurs possibles sont :
  + MAJOR
  + MINOR
  + PATCH

#### Les fonctions

### La classe Module

Une application est subdivisée en modules regroupant chacun un ensemble de fonctionnalités.

#### Les champs

* **id** : LONG unique à chaque module. Cet id est généré par le système.
* **name**: STRING dont la valeur représente le nom du module (exemple : Sourcing). Le nom est obligatoire et est définit par l’utilisateur. Sa taille ne peut excéder 255 caractères.
* **description**: STRING dont la valeur représente la description du module. Ce champ peut être nul.
* **projectVersion**: De type ProjectVersion, il représente la version du projet à laquelle appartient le module. Il ne peut être nul.
* **active**: Boolean indiquant si le module est actif ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Un module inactif est ignoré par le système lors des différentes opérations.

#### Les fonctions

### La classe Functionality

Une fonctionnalité est une subdivision du module. Elle est dédiée au traitement d’une partie bien précise de l’application.

#### Les champs

* **id** : LONG unique à chaque fonctionnalité. Cet id est généré par le système.
* **name**: STRING dont la valeur représente le nom du module (exemple : Sourcing). Le nom est obligatoire et est définit par l’utilisateur. Sa taille ne peut excéder 255 caractères.
* **description**: STRING dont la valeur représente la description du module. Ce champ peut être nul.
* **module**: De type Module, il représente module auquel appartient la fonctionnalité. Il ne peut être nul.
* **active**: Boolean indiquant si la fonctionnalité est active ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Une fonctionnalité inactive est ignorée par le système lors des différentes opérations.

#### Les fonctions

### La classe Issue

#### Les champs

* **id** : LONG unique à chaque Issue. Cet id est généré par le système.
* **name**: STRING dont la valeur représente le nom du module (exemple : Sourcing). Le nom est obligatoire et est définit par l’utilisateur. Sa taille ne peut excéder 255 caractères.
* **description**: STRING dont la valeur représente la description détaillée d’une Issue. Ce champ peut être nul.
* **projectVersion**: De type ProjectVersion, il représente la version du projet à laquelle appartient l’Issue. Il ne peut être nul.
* **createdBy**:
* **creationDate**:
* **owner**:
* **startDate**:
* **endDate**:
* **progress**:
* **catagory**: La catégorie de l’Issue. Il s’agit d’une énumération dont les valeurs possibles sont :
  + FUNCTIONALITY
  + BUG
  + DOC
* **status**: Le statut de l’Issue. Il s’agit d’une énumération dont les valeurs possibles sont :
  + OPEND
  + IN\_PROGRESS
  + RESOLVED
  + CLOSED

#### Les fonctions

### La classe TestScenario

#### Les champs

* **id** : LONG unique à chaque scenario. Cet id est généré par le système.
* **name**: STRING dont la valeur représente le nom du scénario (exemple : Scénario de test de la connexion). Le nom est obligatoire et est définit par l’utilisateur. Sa taille ne peut excéder 255 caractères.
* **description**: STRING dont la valeur représente la description détaillée d’un scénario. Ce champ peut être nul.
* **projectVersion**: De type ProjectVersion, il représente la version du projet à laquelle appartient le scénario. Il ne peut être nul.
* **status**: Le statut de l’Issue. Il s’agit d’une énumération dont les valeurs possibles sont :
  + OPEND
  + IN\_PROGRESS
  + RESOLVED
  + CLOSED

#### Les fonctions

### La classe TestCase

#### Les champs

* **id** : LONG unique à chaque Issue. Cet id est généré par le système.
* **name**: STRING dont la valeur représente le nom du module (exemple : Sourcing). Le nom est obligatoire et est définit par l’utilisateur. Sa taille ne peut excéder 255 caractères.
* **description**: STRING dont la valeur représente la description détaillée d’une Issue. Ce champ peut être nul.
* **scenario**: De type TestScenario, il représente le scenario auquel appartient le cas de test. Il ne peut être nul.
* **preConditions**:
* **postConditions**
* **testDatas :**
* **expectedResults :**
* **weight** : Entier représentant le poids (ou coefficient) du cas de test utilisé dans le calcul du taux de réussite du scénario de test. Sa valeur par défaut est 1.

#### Les fonctions

### La classe TestStep

#### Les champs

* **id** : LONG unique à chaque étape. Cet id est généré par le système.
* **description**: STRING dont la valeur représente la description de l’étape. Ce champ ne peut être nul.
* **case**: De type TestCase, il représente le cas de test auquel appartient l’étape. Il ne peut être nul.
* **active**: Boolean indiquant si l’étape est active ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Une étape inactive est ignorée par le système lors des différentes opérations.
* **position**: La position de l’étape dans la liste des étapes du cas de test.
* **weight** : Entier représentant le poids (ou coefficient) de l’étape de test utilisé dans le calcul du taux de réussite du cas de test. Sa valeur par défaut est 1.

#### Les fonctions

### La classe TestCondition

#### Les champs

* **id** : LONG unique à chaque condition. Cet id est généré par le système.
* **description**: STRING dont la valeur représente la description de la condition. Ce champ ne peut être nul.
* **case**: De type TestCase, il représente le cas de test auquel appartient la condition. Il ne peut être nul.
* **active**: Boolean indiquant si la condition est active ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Une condition inactive est ignorée par le système lors des différentes opérations.
* **position**: La position de la condition dans la liste des conditions du cas de test.
* **category**: La catégorie de la condition. Il s’agit d’une énumération dont les valeurs possibles sont :
  + PRES
  + POST

#### Les fonctions

### La classe TestData

#### Les champs

* **id** : LONG unique à chaque donnée de test. Cet id est généré par le système.
* **name**: STRING dont la valeur représente le nom de la donnée. Ce champ ne peut être nul.
* **value**: STRING représentant la valeur d’une donnée de test. Ce champ ne peut être nul.
* **case**: De type TestCase, il représente le cas de test auquel appartient la donnée de test. Il ne peut être nul.
* **active**: Boolean indiquant si la donnée de test est active ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Une donnée de test inactive est ignorée par le système lors des différentes opérations.
* **position**: La position de la donnée de test dans la liste des données de test de l’étape de test.

#### Les fonctions

### La classe TestExpectedResult

#### Les champs

* **id** : LONG unique à chaque résultat. Cet id est généré par le système.
* **description**: STRING dont la valeur représente la description du résultat attendu. Ce champ ne peut être nul.
* **case**: De type TestCase, il représente le cas de test auquel appartient le résultat. Il ne peut être nul.
* **active**: Boolean indiquant si le résultat est actif ou nom. La valeur par défaut est TRUE. L’utilisateur peut éditer ce champ. Un résultat inactif est ignoré par le système lors des différentes opérations.
* **position**: La position du résultat dans la liste des résultats.

#### Les fonctions