## Base de données

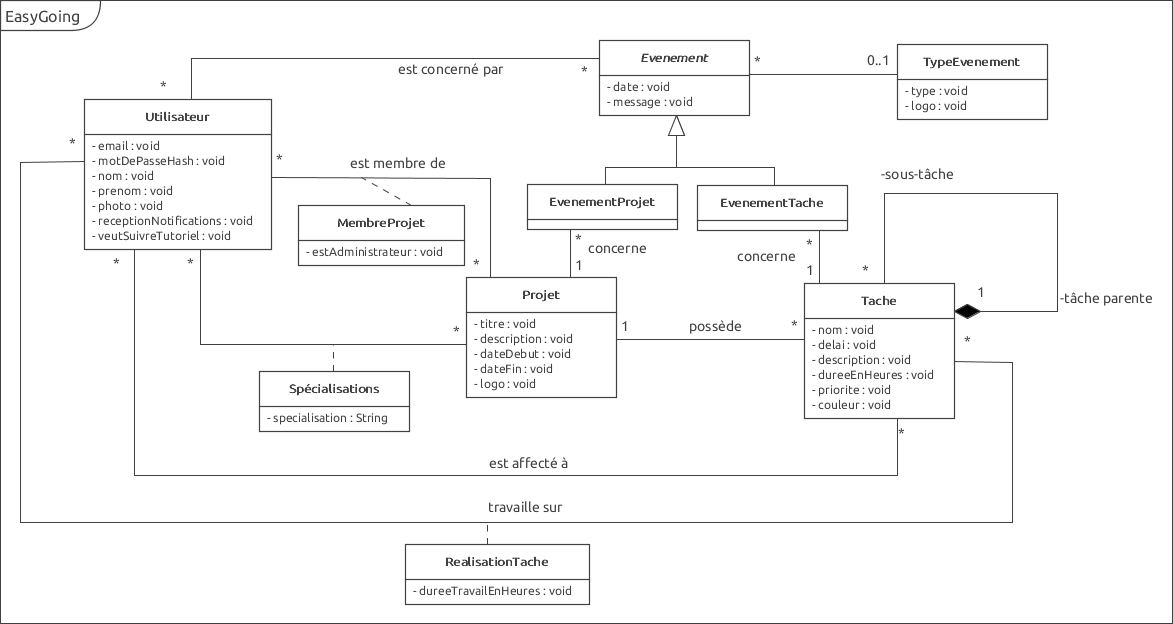
Cette partie du rapport concerne la conception de la base de données.

Voici dans un premier temps notre modèle de données conceptuel (sous forme d’un schéma UML) répondant au mieux au cahier des charges, afin de pouvoir s’y appuyer pour réaliser la base de données relationnel.

Il est à noter qu’il ne s’agit pas du schéma relationnel définitif et qu’il est sous réserve de modifications pour la phase de réalisation, mais les bases sont là.

#### Schéma UML

Voici le schéma UML de notre base de données (modèle de domaine) :



**Refaire le shéma en mettant les types et en retirant les donnes concernant des vues…**

##### Entités

Voici l’explication des entités principales apparaissant sur le modèle. Les données marquées en italique sont des exemples d’utilisation.

|  |  |
| --- | --- |
| **Entité** | **Explication** |
| Projet | Cette entité représente un projet. Chaque projet est caractérisé par :   * Un titre 🡪 *Projet PDG 2015* * Une description 🡪 *Projet de semestre à la HEIG-VD* * Une date de début 🡪 *01.10.2015* * Une date de fin 🡪 *04.01.2016* |
| Tâche | Cette entité représente une tâche d’un projet. Elle est caractérisée par :   * Un nom 🡪 *Faire la vaisselle* * Un délai 🡪 *05.11.2015* * Une description 🡪 *Mettre toutes les assiettes dans le lave-vaisselle* * Une durée en heures 🡪 *La durée estimée de la tâche en heures* * Une priorité  🡪 *Haute* |
| Utilisateur | Cette entité représente un utilisateur. Chaque utilisateur est caractérisé par :   * Un email 🡪 [*raphael.racine@heig-vd.ch*](mailto:raphael.racine@heig-vd.ch) * Un mot de passé (sous forme de hash) 🡪 *ebfh451v65sd1561dfsdf1sdav5s6a* * Un nom 🡪*Raphaël* * Un prénom *🡪 Racine* * Un boolean « réception des notifications » (pour savoir si l’utilisateur désire recevoir des notifications ou pas) 🡪 *True* * Un boolean « veut suivre tutoriel » (pour savoir si l’utilisateur désire avoir le tutoriel qui apparaît) 🡪 *False* |
| Evenement | Cette entité représente un événement (d’une tâche ou d’un projet). Un événement est caractérisé par :   * Une date 🡪 *28.01.2016* * Un message 🡪 *raphaelracine a terminé la tâche X* |

##### Associations

Voici l’explication des différentes associations apparaissant sur le modèle.

|  |  |  |
| --- | --- | --- |
| **Relation** | **Cardinalité** | **Explication** |
| Projet et Utilisateur (MembreProjet) | N - N | Cette relation représente le fait qu’un utilisateur soit membre d’un projet.  Comme il s’agit d’une relation plusieurs à plusieurs, il y aura une entité intermédiaire nommé MembreProjet et cette dernière indiquera si le membre en question est administrateur du projet ou non (booléen « est administrateur ») |
| Projet et Utilisateur P(Spécialisations) | N - N | Cette relation représente le fait qu’un membre puisse avoir plusieurs spécialisations dans un projet. *Par exemple le membre X est spécialisé dans « Java » et dans « Base de données » pour ce projet.*  Il est à noter que ce n’est pas la même association que celle citée précédemment (MembreProjet) car sinon si un membre aurait N spécialisations pour le même projet, il faudrait dire N fois qu’il y a le même membre dans le même projet (1 fois par spécialisation) ce qui serait redondant. |
| Projet et Tâche | 1 - N | Cette relation représente simplement le fait qu’une tâche fasse partie d’un projet. Un projet pouvant avoir plusieurs tâches. |
| Utilisateur et Tâche (affectation) | N - N | Cette relation indique le fait qu’un utilisateur soit affecté à une tâche. Une tâche peut être affectée à plusieurs utilisateurs.  Elle permet de savoir qui doit réaliser une partie de chaque tâche. |
| Utilisateur et Tâche (réalisation) | N – N | Cette relation indique le fait qu’un utilisateur à travailler un certain temps (durée en heures) sur une tâche. *Par exemple le membre X a travaillé 3 heures sur la tâche Y.* |
| Tâche et Tâche | 1 – N | Cette relation représente le fait qu’une tâche peut avoir plusieurs sous-tâches*. Par exemple faire la vaisselle c’est :*   * *Nettoyer les assiettes* * *Nettoyer les verres* |
| Utilisateur et Evénement | N – N | Cette relation représente le fait qu’un utilisateur soit concerné par un événement. Un événement peut concerner plusieurs utilisateurs. |
| Evénement et TypeEvénement | N – 1 | Cette relation indique qu’un événement possède un type. |
| Evénement et EvénementProjet | Héritage | L’entité EvenementProjet concerne un projet et elle hérite de l’entité Evénement.  Elle a comme caractéristique supplémentaire le fait qu’elle concerne un projet. |
| Evénement et EvénementTache | Héritage | L’entité EvenementTache concerne une tâche et elle hérite de l’entité Evénement.  Elle a comme caractéristique supplémentaire le fait qu’elle concerne une tâche. |
| Projet et EvénementProjet | 1 – N | Cette relation représente simplement le fait qu’un événement de projet est lié à un projet. |
| Tâche et EvénementTâche | 1 – N | Cette relation représente simplement le fait qu’un événement de tâche est lié à une tâche. |

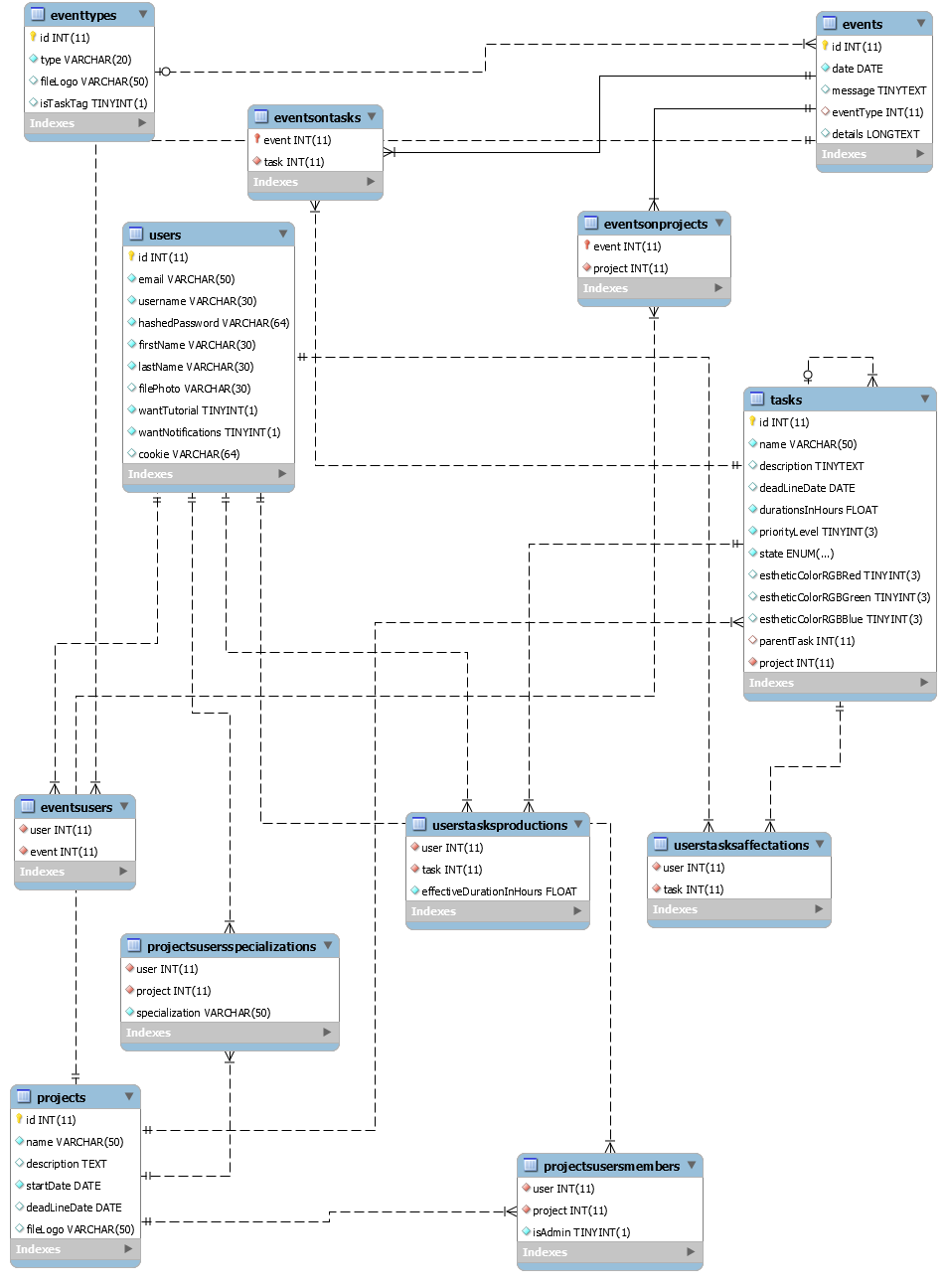
#### Contraintes d’intégrité

Voici les contraintes d’intégrités principales liées à notre base de données :

* Membre projet : Un utilisateur ne peut pas être 2 fois membre d’un même projet, et il doit y avoir au moins un administrateur par projet
* Affectation de tâche : Un utilisateur ne peut pas être affecté à une tâche s’il n’est pas membre du projet dans lequel la tâche est inscrite
* Réalisation de tâche : Un utilisateur ne peut pas réaliser une tâche s’il n’y est pas affecté
* Sous tâche : Une tâche qui a déjà une tâche parente ne peut pas avoir de sous-tâche. Autrement dit, on s’arrête à un seul niveau de sous-tâche

#### Schéma relationnel

Voici dans un premier temps le schéma relationnel (créé à l’aide du logiciel WorkBench) (de manière global, il sera ensuite expliqué par partie).



#### Tables

Voici une description des différentes tables de la base de données, regroupées par catégories.

##### Utilisateurs

Ici est expliquée la tables concernant les utilisateurs :

* Users

|  |  |  |
| --- | --- | --- |
|  |  |  |

##### Projets

Ici sont expliquées les tables concernant les projets :

* Projects
* ProjectsUsersSpecializations
* ProjectsUsersMembers

Voici comment elles sont reliées entre elles :

**Mettre le diagramme**

|  |  |  |
| --- | --- | --- |
|  |  |  |

##### Tâches

Ici sont expliquées les tables concernant les tâches :

* Tasks
* UsersTasksAffectations
* UsersTasksProductions

Voici comment elles sont reliées entre elles :

**Mettre le diagramme**

|  |  |  |
| --- | --- | --- |
|  |  |  |

##### Evénements

Ici sont expliquées les tables concernant les événements :

* Events
* EventType
* EventsOnProjects
* EventsOnTasks
* EventsUsers

Voici comment elles sont reliées entre elles :

**Mettre le diagramme**

|  |  |  |
| --- | --- | --- |
|  |  |  |

#### Vues

Voici les différentes vues de notre base de données utilisées dans l’application.

**Expliquer les vues**

#### Procédures, fonctions et triggers

Voici les procédures, fonctions et triggers appartenant à la base de données.

**Expliquer**

## Tutoriel (à mettre dans la partie technique du rapport)

### Réalisation

Voici comment nous avons réalisé la partie des tutoriaux qui s’affichent sur les pages principales de l’application, sous forme de petites fenêtres comme celle-ci :

**Mettre un exemple…**

#### Partie serveur

* **Contrôleur**
* **Modèle**

#### Partie client

* **Javascript (tuto.js, loadTutorial.js)**
* **Utilisation dans une page**