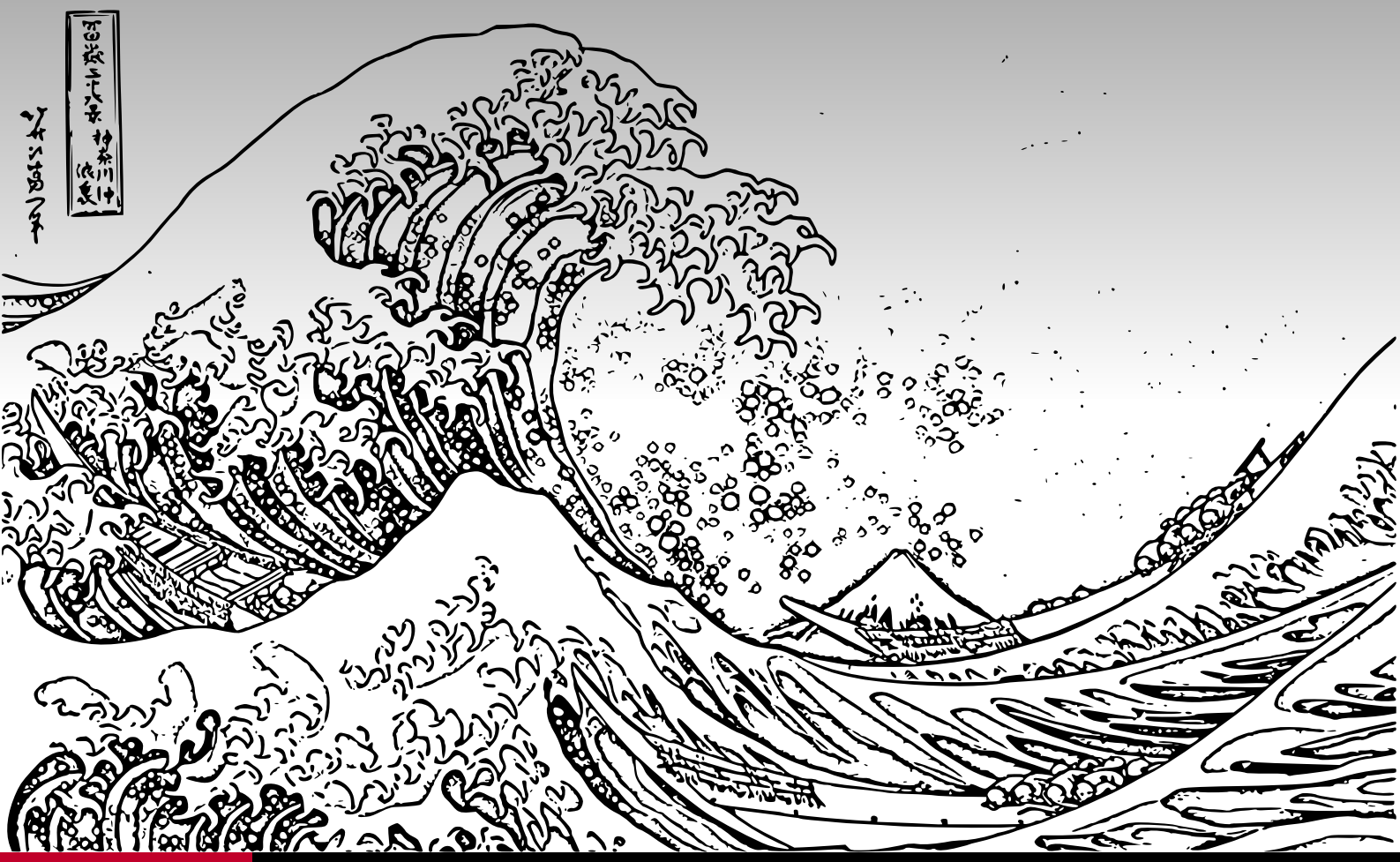
A ho'e tatou !

2023



le cnam

A project by Vaanaiki BROTHERSON

Project L3 - Proof of Concept - Documentation

Va'a Training Tracking Application - A Hoe Tatou

Client: Va'a Team Coach
Delivery Date: UNDETERMINED
@author : Vaanaiki BROTHERSON aka Naiki
@date : September 2023


@github : https://github.com/naikibro/a-hoe-tatou
@LinkedIn : www.linkedin.com/in/naiki-brotherson987

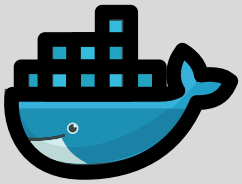
Table of content

This document is the functional and technical documentation for the **A hoe tatout** project You will find here the Table of content of the differents parts of this project :

Concepts, Methods & Implemented Technologies

Tools, Concepts, and Frameworks

Laravel PHP 8.1.2-1ubuntu2.14 (CLI): Laravel Documentation Model-View-Controller (MVC): Introduction to Laravel and MVC Framework Docker: Docker Installation
Composer: Composer Download
MariaDB Server: MariaDB Documentation
SQL: SQL on Wikipedia

Agile project management

Project tracking will be done using Agile methods and more particularly Scrum, Roadmap and changelog can be followed on the project's ClickUp

le cnam

Agile project management

The advancement of the project can be followed and monitored on the following clickUp link

https://app.clickup.com/9016001649/v/li/901600008311

About the agile methods

We will use the SCRUM method to track the progress of the project
Every User-story and User-case shall be listed as a ticket in the clickup project

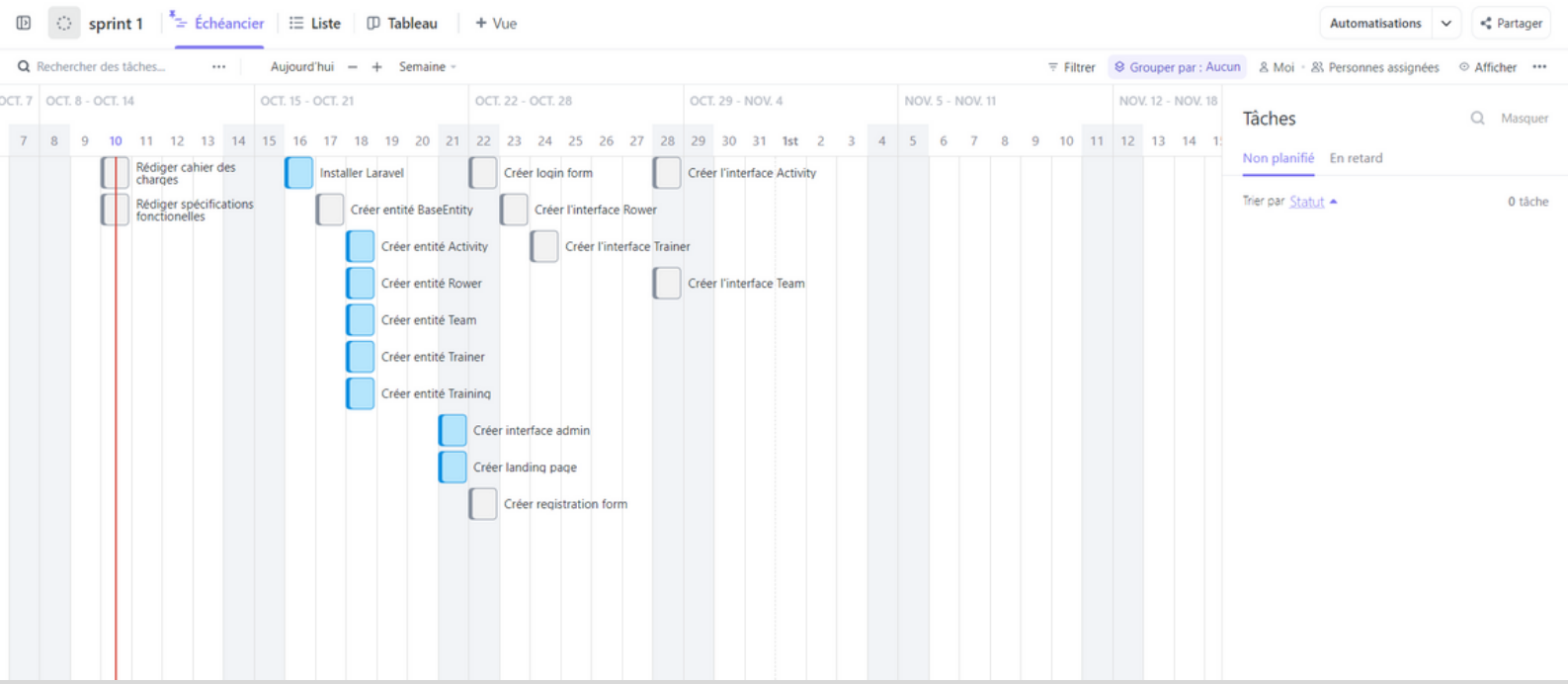
US and UC will be sorted as Epics and then trimmed down in tickets of features
A sprint will last for 14 days
An intesprint shall be organized on the 8th day, to ponder the workload again and modify the workload accordingly

About the Epics and the features

An EPIC is a thematic collection of features that together form a larger, cohesive unit of work within a project or product development process. Features within an EPIC should ideally adhere to the INVEST ruleset when defining user stories. INVEST is an acronym that stands for:

1. Independent: Each feature within an EPIC should be independent of others as much as possible. This means that they can be developed, tested, and deployed without strict dependencies on other features. Independence allows for flexibility in prioritization and execution.
2. Negotiable: Features should be open to negotiation and discussion. They should not be set in stone but should evolve based on feedback and changing requirements. This flexibility ensures that the team can adapt to new information and priorities.
3. Valuable: Every feature should provide value to the end users or stakeholders. It should solve a real problem or address a genuine need. Ensuring that features deliver value helps in prioritizing and justifying their inclusion in the EPIC.
4. Estimable: Features should be clear and well-defined enough that the development team can estimate the effort required to implement them. This helps in planning and allocating resources effectively.
5. Small: Features should be kept small and manageable. Smaller features are easier to develop, test, and deploy. They also allow for more frequent releases, which can lead to faster feedback and improved product quality.
6. Testable: Each feature should have clear acceptance criteria that define when it is considered complete and working as expected. This ensures that the team and stakeholders have a shared understanding of what success looks like.

Following the INVEST ruleset when defining user stories within an EPIC helps teams maintain a high level of agility and adaptability. It encourages the creation of well-structured, valuable, and manageable features that can be developed independently, tested effectively, and ultimately provide meaningful benefits to users and stakeholders.

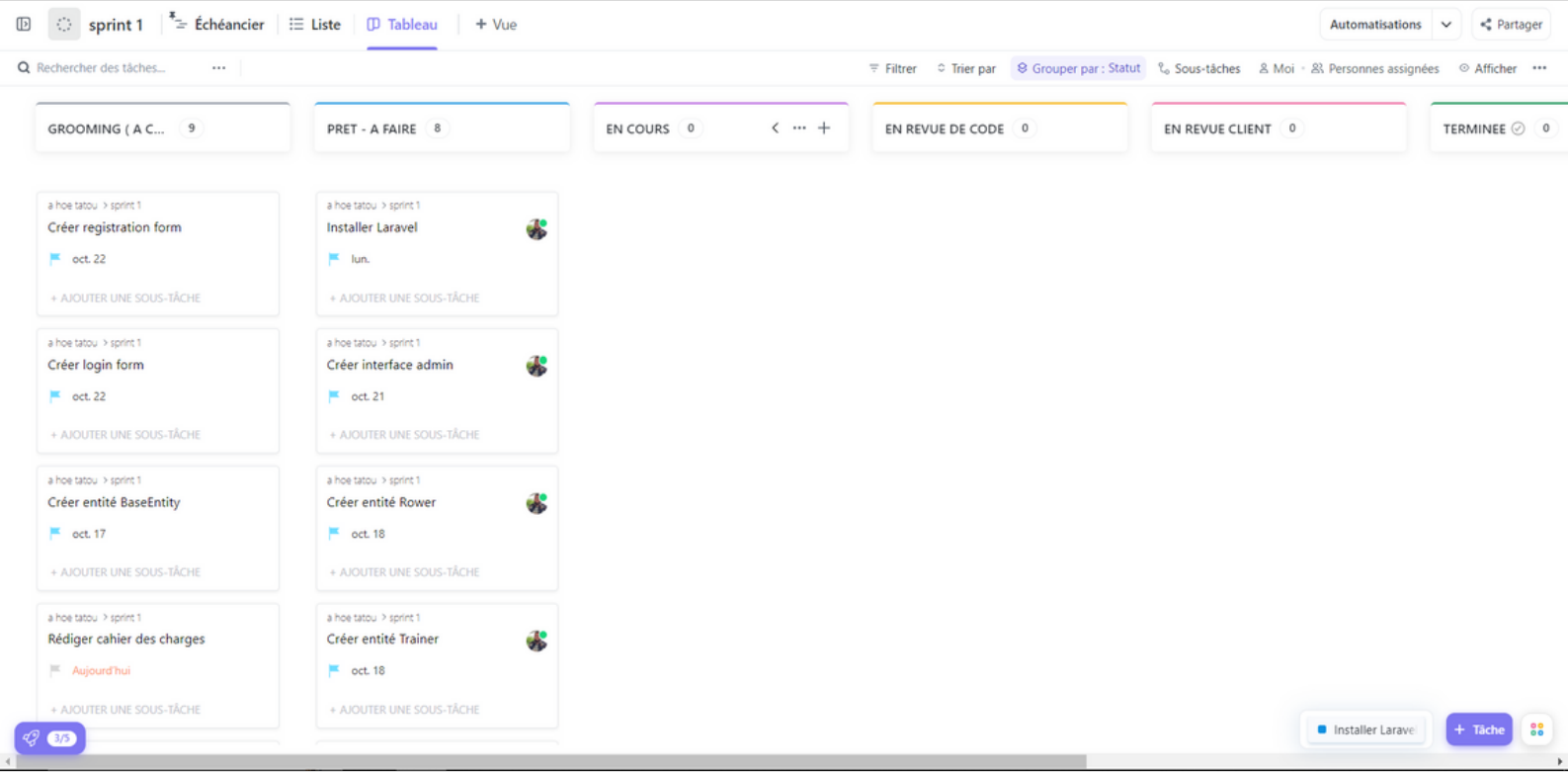


Backlog and table of implementation

We are gonna use backlogging and milestones in this project to track the advancement in time of our project.

Every card and tasks visible here can be seen in the clickup
Cards and developpement of features will be done in sprints

A sprint typically lasts for 2 weeks



Roadmap (with versions)

V0.0.0 - Genesis

- 0.0.0 : setup the github and docker environement
- 0.0.1 : setup the laravel dev environement
- 0.0.2 : setup the laravel production environement
- 0.0.3 : setup the CI/CD pipeline
- 0.0.4 : initialisation of project
- 0.0.5 : create baseClass for global attributes (createdAt and updatedAt)
- 0.0.6 : create whole database (schemas and fakers)
- 0.0.7 : create an admin dashboard
- 0.0.8 : create authentication system for admin
- 0.0.9 : create User related system
- 0.0.10 : create front office for Users
- 0.0.11 : create authentication system for user
- 0.0.12 : create User handling of account (CRUD)
- 0.0.13 : test code
- 0.0.14 : fix bugs
- 0.0.15 : refactor code

V0.1.0 - Setting up the skeleton

- 0.1.0 : create all classes of entities
- 0.1.1 : create admin CRUD controllers for all classes
- 0.1.2 : create Mailer system
- 0.1.3 : create Messaging system (on premises)
- 0.1.4 : test code
- 0.1.5 : fix bugs
- 0.1.6 : refactor code

V0.2.0 - Setting up the front office

- 0.2.0 : create all front pages for User
- 0.2.1 : custom the global appearance to match graphical identity
- 0.2.2 : custom the authentication system appearance
- 0.2.3 : fixing routing for users
- 0.2.4 : fixing routing for admins

V0.3.0 - Implementing relations between entities

- 0.3.0 : create CRUD for User
- 0.3.1 : create CRUD for Trainer
- 0.3.2 : create CRUD for Teams
- 0.3.3 : create CRUD for Trainings
- 0.3.4 : create CRUD for Activities
- 0.3.5 : create Notification system
- 0.3.6 : create Reminder system

Roadmap (with versions)

V0.4.0 - Time management features and inscriptions

- 0.4.0 : create Calendar vue for Trainings
- 0.4.1 : create Calendar vue for Activity
- 0.4.2 : create global Calendar vue
- 0.4.3 : implement auto-invitations for Trainings
- 0.4.4 : implement auto-invitations for Activities
- 0.4.5 : implement automatic reminders for Trainings
- 0.4.6 : implement automatic reminders for Activities

V1.0.0 - Minimum viable product

A user can now :

- Own or create an account
- log into the app
- join teams
- join trainings
- join activities
- chat on the app
- receive notifications and emails
- track his trainings and activites on the calendar
- become a Trainer

A Trainer can now :

- Create or modify his teams
- Create Trainings
- Create Activities
- Message his team mates
- Invite mates to Trainings
- Invite mates to Activities

Version control

The different versions of the project are to be stored in the following github repository

https://github.com/naikibro/a-hoe-tatou

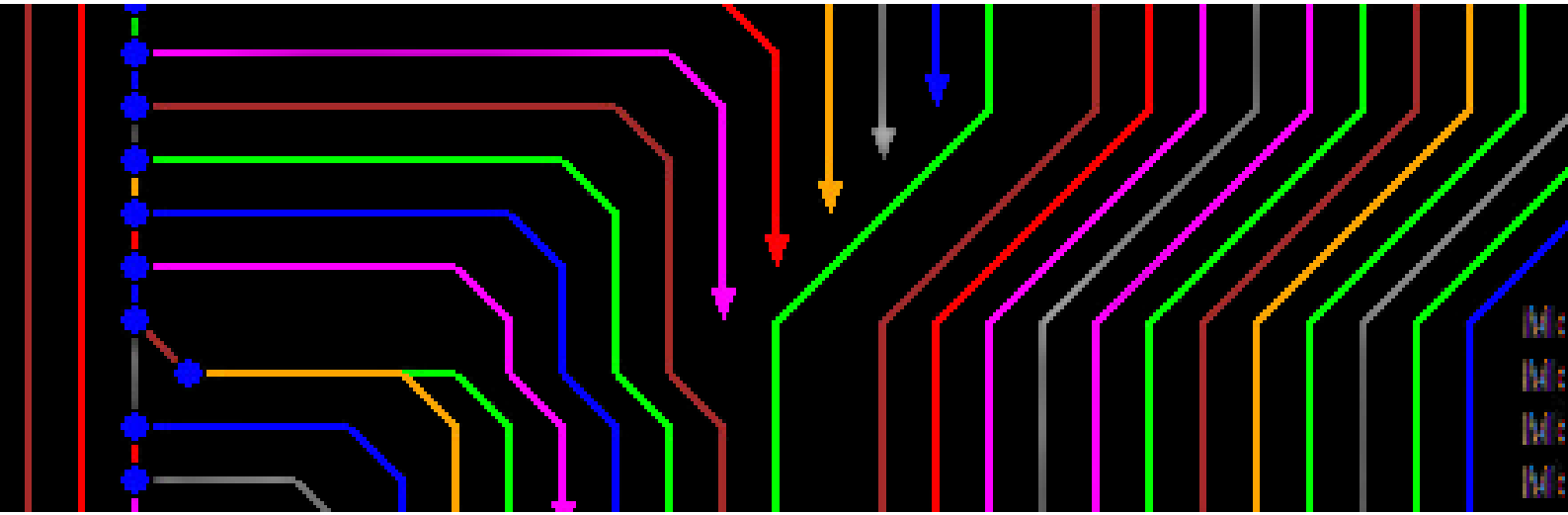
About the version control

We will use GIT versionning inside Github to manage the versions of our project

Developement of features shall be made inside specific branches on the developper computer and then merged, tested and approved to the main branch of the distant repository

Why we are using VCS

1. Collaboration: Version control systems (VCS) facilitate collaborative work by allowing team members to work on their own branches of the project and merge changes seamlessly.
2. History Tracking: VCS keeps a detailed history of all changes made to the project, making it easy to identify when and why specific modifications were introduced.
3. Error Recovery: If an issue or bug is introduced, version control makes it possible to roll back to a previous, stable state of the project, reducing the impact of errors.
4. Branching and Merging: Developers can create branches to work on specific features or fixes independently and later merge them back into the main codebase, ensuring a structured and organized development process.
5. Parallel Development: Teams can work on different aspects of the project simultaneously without interfering with each other's work.
6. Documentation: Version control systems often include tools for documenting changes, which can be helpful for understanding why specific decisions were made.



Graphical identity and mockups

Project tracking will be done using Agile methods and more particularly Scrum, Roadmap and changelog can be followed on the project's ClickUp

le cnam

Graphical identity

The logo

Color palette

Mockups

A graphical mockup of the desired project

https://xd.adobe.com/view/bb7ff8aa-68b8-483f-8071-c75a77806974-ee2c/?fullscreen&hints=off

ONE APP FOR ALL YOUR VA'A NEEDS !

User definitions

User stories and use case diagrams

cf Jira board :
https://naikibro.atlassian.net/jira/software/projects/HT/boards/2
User stories are dissecated into Jira cards You will find here all of the user stories of our app

le cnam

Roles Definition

Passive Users

Visitor: 

- Does not have an account.
- Is not logged in.
- Is always redirected to a registration + login page.

Unregistered Rower: 

- Has an account.
- Is not registered in any team.

Active Users

Registered Rower: 

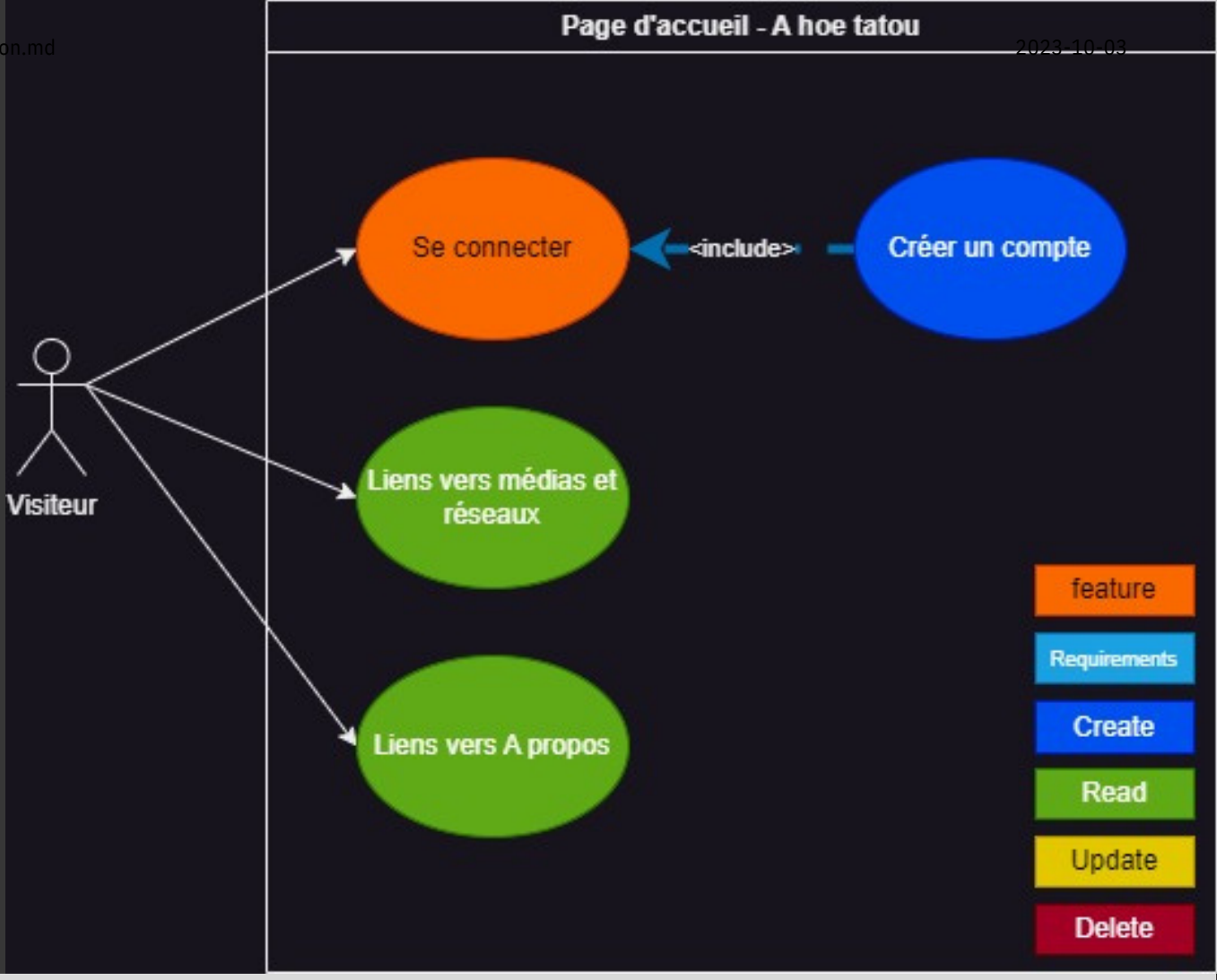
- Has an account.
- Is registered in at least one crew and/or team.

Trainer: 

- Has an account.
- Is registered in at least one crew and/or team.
- Is registered in the trainers' registry.

CAUTION!: A user can be both a Rower and a Trainer for multiple crews (roles are specific to crews).

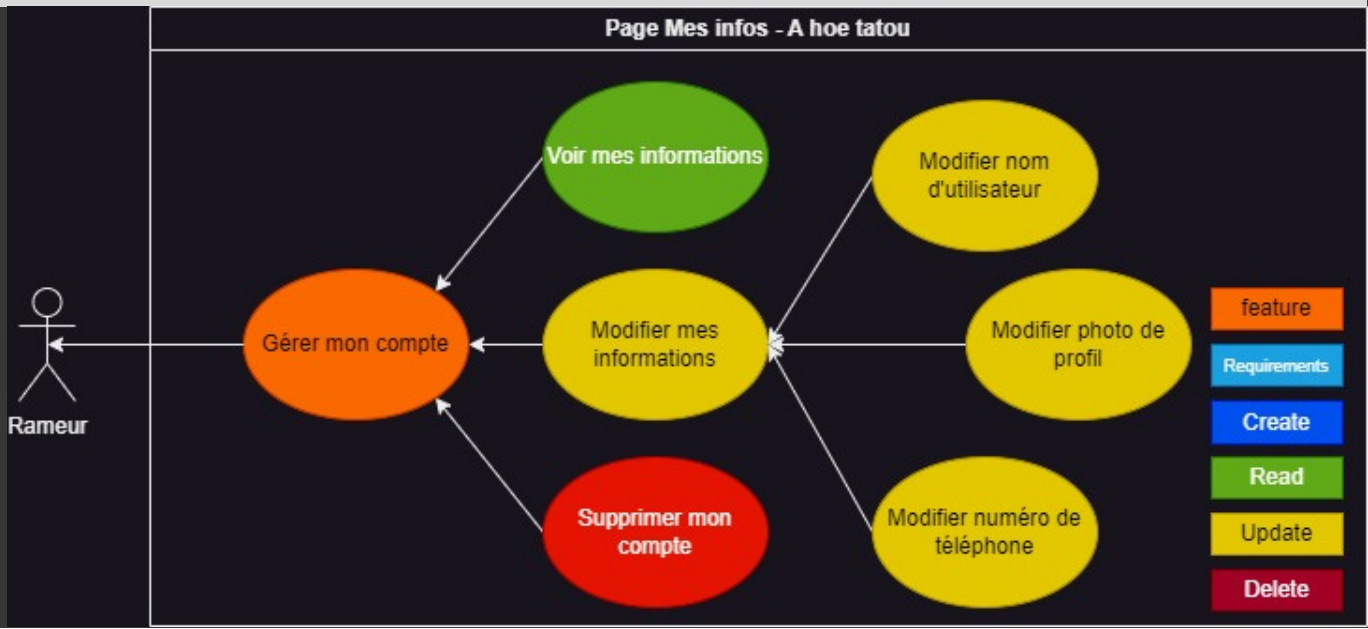
Visitor



USV1 - LandingPage
USV2 - CreateAccount
USV3 - Login

USR1 - Login
USR2 - ModifyAccount
USR3 - RemoveAccount

Rower



Teams

USRT1 - GetTeam
USRT2 - CreateTeam
USRT3 - ModifyTeam
USRT4 - LeaveTeam
USTT1 - RemoveTeam
USRT5 - SendMessageToGroupChat
USRT6 - SendMessageToGroupChat

Teams Inscriptions

USST1 - addUserToTeam
USST2 - RemoveUserFromTeam

Trainings

Trainings CRUD

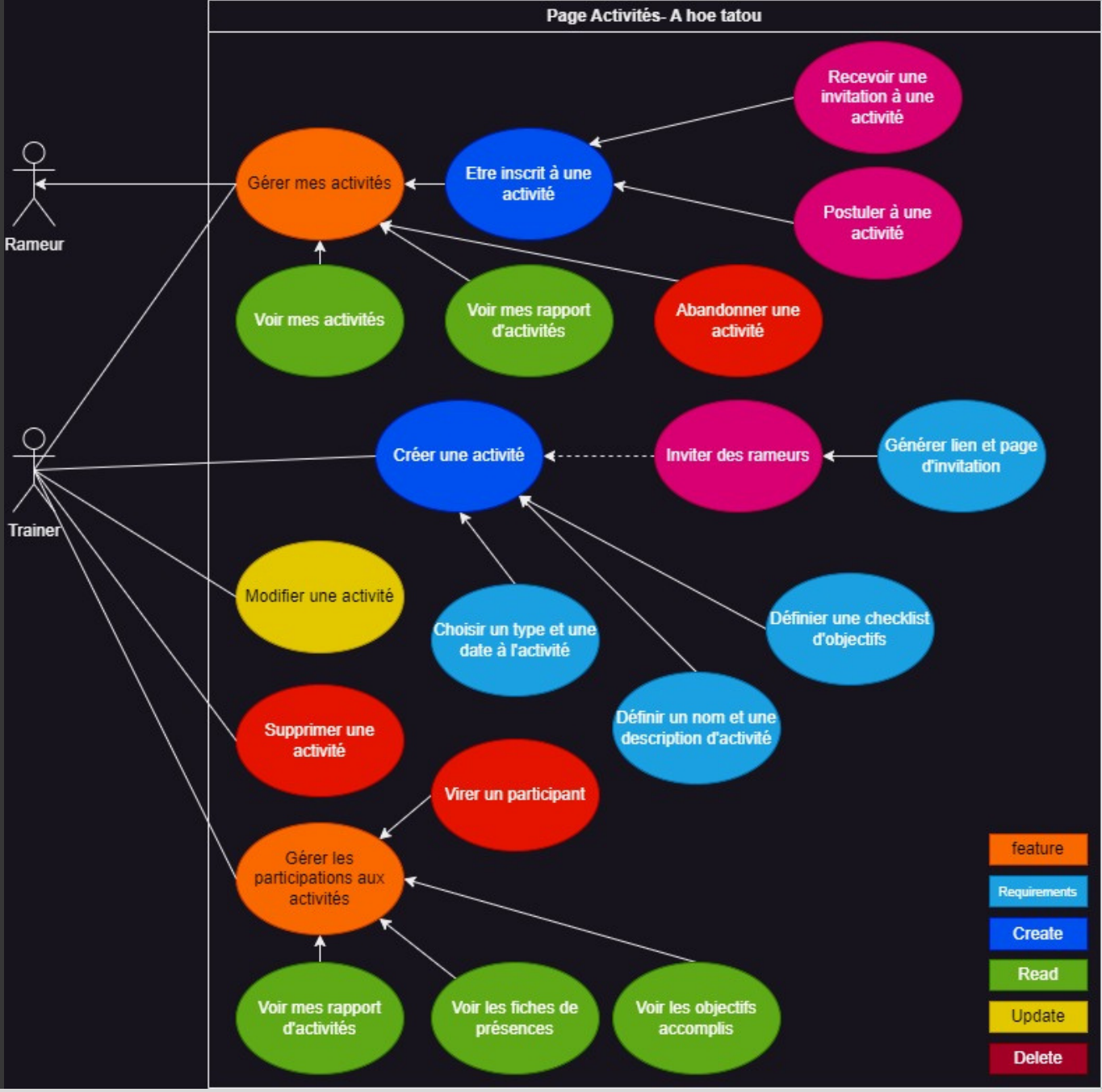
USTTr1 - GetTraining
USTTr2 - CreateTraining
USTTr3 - ModifyTraining
USTTr4 - RemoveTraining

USRTr1 - GetTraining
USRTr2 - InviteRower
USSTr1 - GetPerformance

Trainings Inscriptions

USRTr3 -EnrollForTraining
USRTr4 - AbandonTraining
USRTr5 - ChoosePosition

Activities



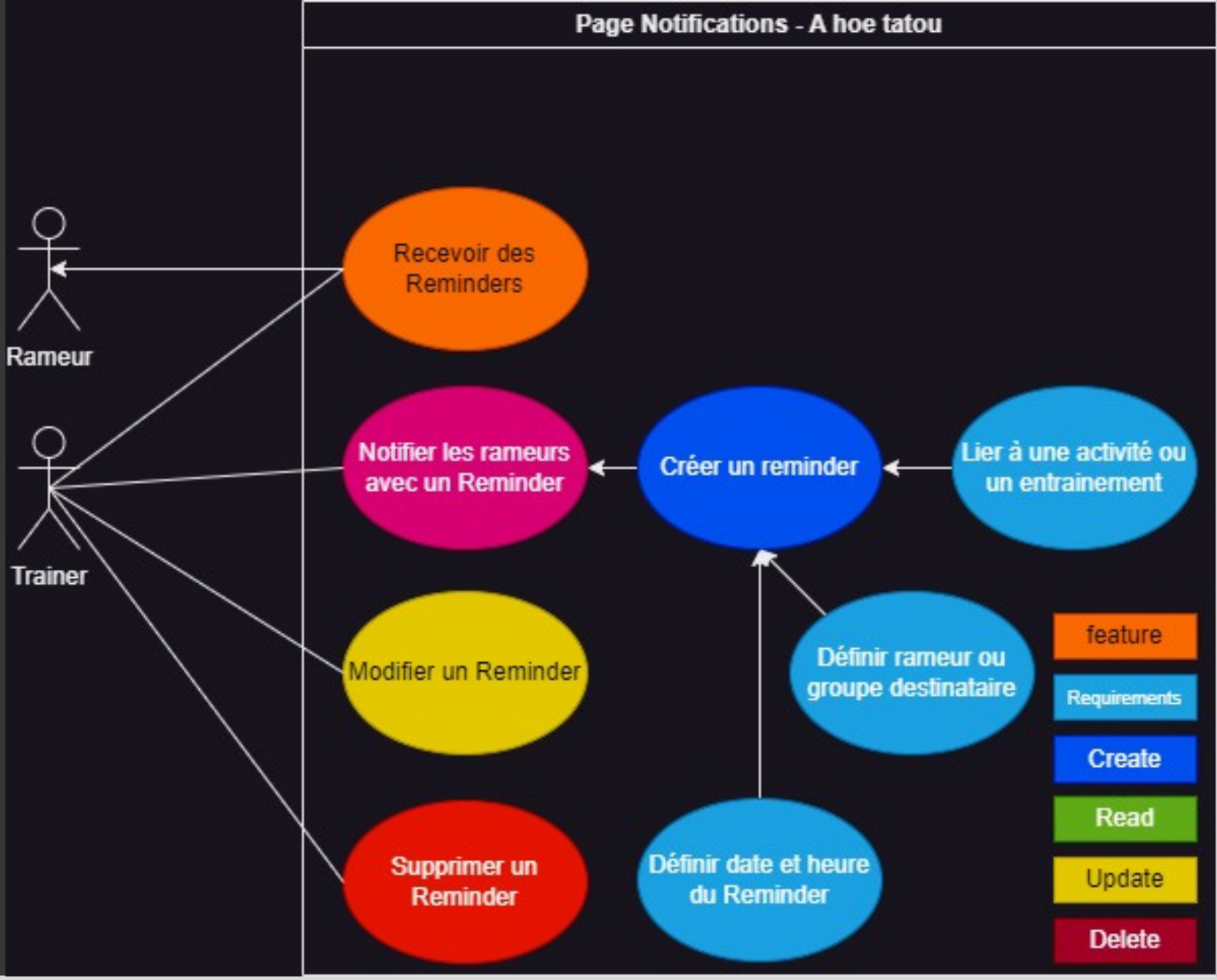
Activity CRUD

user-story:Trainer - GetActivity
user-story:Trainer - SetActivity
user-story:Trainer - ModifyActivity
user-story:Trainer - RemoveActivity

Activity Inscriptions

user-story:Rower - EnrollForActivity
user-story:Rower - AbandonActivity

Reminders



Reminders

user-story:Trainer - GetReminder
user-story:Trainer - SetReminder
user-story:Trainer - ModifyReminder
user-story:Trainer - RemoveReminder

user-story:Rower - GetReminder

user-story:Trainer - AlertBenchers
user-story:Trainer - AddRower
user-story:Trainer - InviteRower

System

user-story:User - TurnOnNotifications
user-story:User - TurnOffNotifications

user-story:Practice - AutoFillVaa
user-story:Practice - EmergencyAutoFillVaa

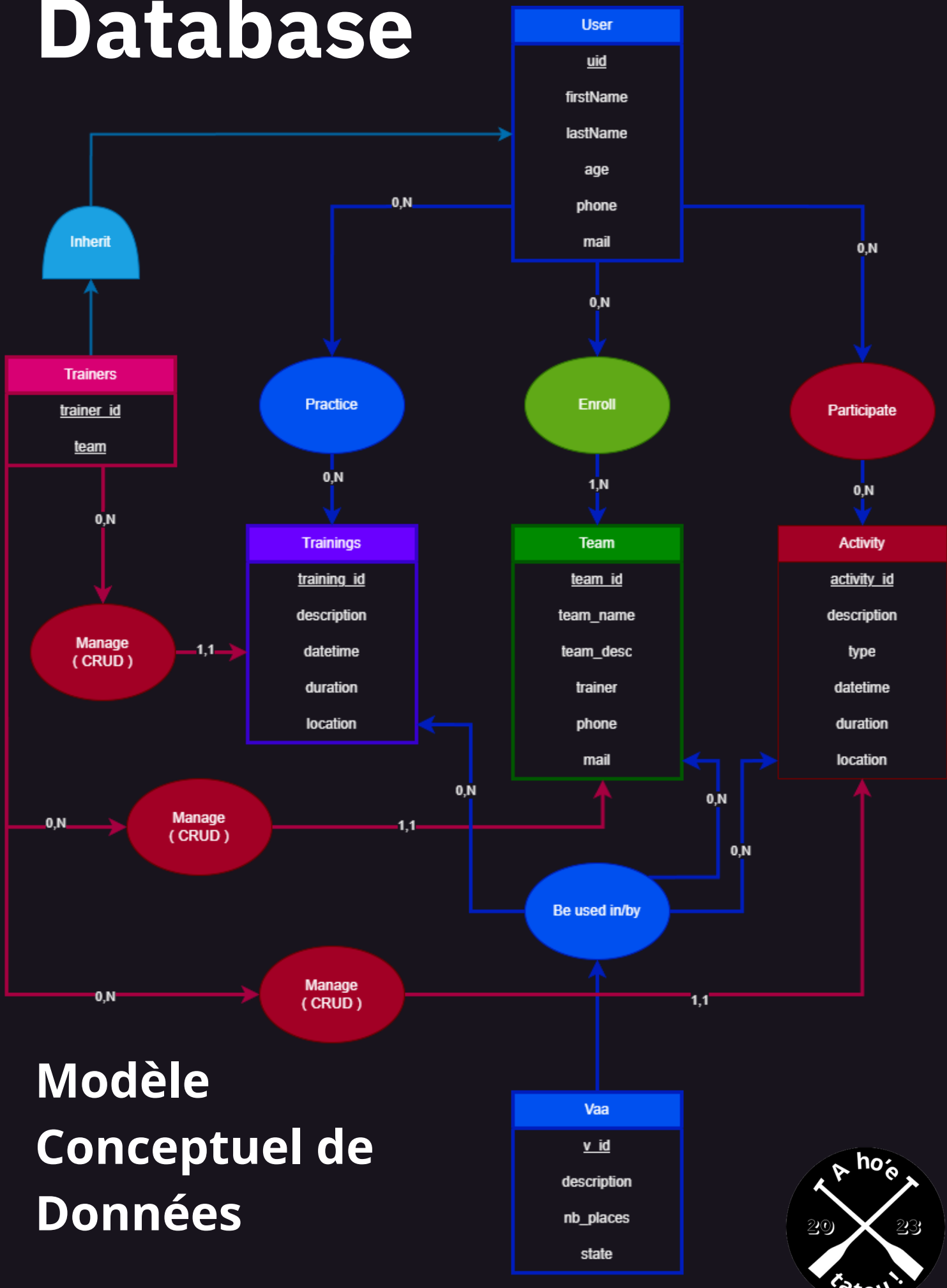
Database schemas

You will find here the MCD and MPD of the database

le cnam

DBMS : MariaDB
Query language : SQL
diagram tools : SQL flow, draw.io

Database



Modèle Conceptuel de Données

A ho'e tatou ! 2023

Modèle logique de données