**Subject**

**General elements:**

* ge1 nice user interface
* ge2 a chat
* ge3 real-time multiplayer online games

**Dev-ops**:

* do1 launch by a single call to: docker-compose up --build
* do2 any credentials, API keys, env variables etc... must be saved locally in a .env file
* do3 any credentials, API keys, env variables etc... must be ignored by git.

**Front-end design:**

* fe1 typeScript framework of your choice
* fe2 latest stable version of every library or framework
* fe3 single-page application (able to use the Back and Forward buttons)
* fe4 compatible with the latest stable version of Google Chrome
* fe5 compatible with the latest stable version of one additional web browser
* fe6 no unhandled errors when browsing
* fe7 no warnings when browsing
* fe8 protected against SQL injections.

**Back-end design**

* be1 written in NestJS
* be2 latest stable version of every library or framework
* be3 some kind of server-side validation for forms and any user input.

**Back-end database**

* db1 PostgreSQL database
* db2 Any password stored in the database must be hashed
* db3 strong password hashing algorithm

**Game design:**

* gd1 play a live Pong game versus another player
* gd2 matchmaking system (via a queue)
* gd3 3D or original look and feel
* gd4 default version or customizable options
* gd5 responsive
* gd6 robust to network issues (unexpected disconnection or lag)

**Front-end functionalities:**

* ff1 login using the OAuth system of 42 intranet
* ff2 user choose a unique name that will be displayed on the website.
* ff3 user upload an avatar (if not a default one)
* ff4 two-factor authentication
* ff5 set friends and check friend status
* ff6 User status: online, offline, in a game, and so forth)
* ff7 Stats displayed on the user profile
* ff 8 Stats values: wins and losses, ladder level, achievements, and so forth) have to be .
* ff9 User has match history
* ff10 Mach history available for everybody
* ff11 Mach history values: 1v1 games, ladder, and any- thing else useful.

**Chat functionalities:**

* cf1 user should be able to create channels
* cf2 channels are: public, or private, or protected by a password
* cf3 user should be able to send direct messages
* cf4 user should be able to block other users (messages)
* cf5 channel has owner: the creator user
* cf6 channel owner can set a password
* cf7 channel owner can change the password
* cf8 channel owner can remove the use of password
* cf9 channel has administrator
* cf10 owner is administrator
* cf11 administrator can set news administrators
* cf12 administrators can kick, ban or temporal mute users (not the owner)
* cf13 new game invitation from chat available
* cf14 other players profile form chat available

**Evaluation sheet**

**Introduction**

* be polite, discuss until the evaluator agrees you are right

**Guidelines**

* use git repository
* no segfault or uncontrolled finish
* no memory leaks

**General instructions**

**Preliminary tests**

* Any credentials, API keys, environment variables must be set inside a .env file during the evaluation. In case any credentials, API keys are available in the git repository and outside of the .env file created during the evaluation, the evaluation stop and the mark is 0. (do2 do3)
* Ensure the docker compose file is at the root of the repository.
* Run the "docker-compose up --build" command. (do1)
* Since the rating of this project is more flexible, do not stop the evaluation process unless you encounter a 500 error, a crash, or anything that actually doesn't work within the project scope.

**Backend**

* The backend must be developed using the NestJS framework. (be1)
* The database must be a PostgreSQL database. (db1)
* During the whole evaluation process, there must be no unhandled warning or error. (fe6)

**Frontend**

* The frontend must be done using a TypeScript framework. (fe1)
* Any TypeScript/JavaScript library is allowed. (fe1)
* During the whole evaluation process, there must be no unhandled warning or error. (fe6)

**Basic checks**

* The website is available at the address chosen by the students.
* The user can login using the 42 intranet OAuth feature. (ff1)
* When logged for the first time, the user is prompted to add information to their account (display name/nickname, avatar, and so forth).
* If not logged, the user has access to only little or no information and is prompted to sign in.
* The website is a Single Page Application. The user can use the "Back" and "Forward" buttons of the web browser. (fe3)
* You can browse the website using the latest version of Chrome and one additionnal browser without encountering any problems or errors. (fe4 fe5)

**The website**

**Security concerns**

* Ensure that the website is secured. (ff1)
* Check the database to verify that passwords are hashed. (db2)
* Check the server for server-side validation/sanitization on forms and any user input. If this isn't done, the evaluation ends now. (f8, be3)

**User profile - private**

* When logged in, the user has access to their profile
* the user can edit their information. (ff1 ff2 ff3 ff4 ff5)
* they can change their nickname (which must be unique) (ff2)
* they can change their avatar (which is a default avatar if not set). (ff3)

**User profile - public**

* Users can see the profile of other users. (ff5)
* A profile contains basic informations such as their nickname, their avatar, or a button to add them as friends. (ff1 ff2 ff3 ff4 ff5)
* User can block other users. This means they won't receive private messages from the accounts they blocked nor view their messages in public/private channels. (cf4)

**Friend interface**

* The user has access to a friends interface, where they can see their friends and their status (offline/online/in a game/and so forth).
* They also have access to basics informations about them (name/nickname, avatar and so forth).

**2FA**

* The user can enable/disable 2FA (two-factor authentication). (ff4)
* If enabled, they must pass it in order to sign in. (ff4)
* For example, 2FA can use Google Authenticator, a text message, an email, and so forth. (ff4)

**Chat interface**

**Join/leave channels**

* A logged in user can access the website chat service. (ge2)
* Joining/leaving channels is a manual action. For example, this means it must not be done on logout (the user has to click a "Leave channel" button or something else). cf2
* The user can join channels (that can be already created) to have a chat. cf2
* Some of them can be password-protected. If so, the user has to enter the correct password in order to join the channel. cf2, cf6, cf7 cf8

**Chat usage**

* The users can chat.
* Messages must be sent/received instantly.
* If the user blocked another user, the messages from the blocked person must be hidden.
* The user can access the user profile of other players from the chat interface and also invite them for a Pong duel. cf13

**Creating channels**

* The user can create new channels. cf1
* The channel creator is set as the channel owner and has basic moderations rights (ban/mute users, add a password to protect the channel, set new administrators, and so forth). (cf5 a cf12)

**Channel roles**

* A user who is an owner of a channel can kick, ban, mute other users and the channel administrators. cf 12
* A user who is an administrator of a channel can kick, ban mute other users, but not the channel owners. cf12

**The game**

**Matchmaking system**

* When logged in, the user has access to a matchmaking system so they can play Pong 1v1 games versus other players on the website. (gd2 gd1)
* When they get matched, a new game is loaded and the two users can start playing. (gd1)

**Gameplay**

* The game itself must be playable and respect the original Pong game. (gd3)
* The controls must be intuitive or correctly explained (with some rules or manual).
* When a game is over, either a kind of end-game screen is displayed or the game page just exits.

**Spectator mode**

* There is a spectator mode.
* The user can watch any live games. They can be accessed through the chat interface or the friend interface.
* There can also be a page dedicated to live games from which the user can access any of them.

**Lags & disconnects**

* Unexpected disconnections and lags have to be handled. (gd6)
* The game and the website must not crash when a user is experiencing lags or is disconnected. (gd6)
* Handling such issues in an efficient way is appreciated but not mandatory:
  + Pause the game for a defined duration.
  + Disconnected users can reconnect.
  + Lagging users can catch up to the match. And so forth.

Any solution is acceptable. The only requirement is: the game should not crash.(gd6)

**Additional features**

* The user can enjoy extra features such as power-ups, different maps, achievements, and so forth. (gd4 gd3)