ToDo on Symfony 6 Framework : General guidelines

1. Architecture

a) Application heart

ToDo App integrates the Symfony 6 framework and follows its principles. **To updates its functionalities**, you must respect the architecture of the site to keep the source code intelligible and accessible by any developper.

To add attributes to an entity, you can either use the maker command from the maker bundle (link) or adapt yourself the entity file, repository and formType (follow the existing models of the application).
In the same spirit, **to create a new entity**, you can either use the maker command from the maker bundle or create yourself the entity file, repository and formType (follow the existing models of the application).

Business logic (namespace : App/Controller): any **new functionality** our **update of functionalities** must be implemented on distinct controllers. In its current state, ToDo app is animated by two entities: Task and User. The Task functionalities must be written in the TaskController, the User functionalities must be written in the UserController. **Make sure to annotate any new function with an url and a name to make it known to the router and comment them to sumup what it does.**

Repository (namespace: App/Repository): In the controllers, you have access to doctrine methods to access your data. If the methods are not enough to get the data sample you want, **specific requests must be written**. Following the controller's example, Users and Tasks have their own repository. **If you want to retrieve Tasks, write the logic in the TaskRepository (user in UserRepository)**.
Repositories are accessible from the controllers. Follow the syntax : $this->em->getRepository(ClassWanted::class)->methodNeeded();

The only logic that escapes those rules are the security specifications described in the first part with specific controllers.

b) Test environment

At the base of Tests file, you'll find the HTML coverage reports and the entity and controller files :

- unit test are written in entityTest files
- functional tests are written in entityControllerTest files

2. Quality process

To ensure the best performance and collaboration between developers, development process must be applied.

When adding a new functionality or entity to the App, the following steps must be followed :

1. Create the tests that will confirm the viability of the changes

    - for an entity modification, unit tests are to be written in entityTest files (namespace App\Tests\Entity). Those will ensure the effectiveness of attributes constraints

- for a new business logic, in the entityControllerTest files (App\Tests\Controller). Add as much assertions as it is needed to validate the viability of the function and the management of the possible errors. Assertions must check the success of a response, the effective changes in the dataBase and the following results in the final view sent to the user.
  - Make sure the method's name start with "test…" so phpUnit identifies it.
  - Run the test to confirm their failure. They will be our checkpoint of success for the changes added to the App

2. Add the actual application changes to the App

   - in the /Src files respecting the place of each item (entity, logic, database communication). Make sure to respect performance and collaborative guidelines :
     - keep it simple and avoid long methods
     - apply the same indentation you'll find in the existing files
     - don't write unused codes or variable
     - Do not rewrite an existing logic. If needed, factorize

   - Run the test to check the success of each assertions. Once the test are successful, you can consider the new functionality ready to be pushed

3. After merging, check the updated results of the updated Codacy audit to maintain an acceptable lovel of performance. The audit will guide you into the changes to be done if mistakes have been made

You can chose to exclude from the test code or functionalities that are not necessary or critical for the business logic. But beyond this critical thinking, ensure a test coverage of the app of at least 70% as proof for the App viability. You can find this coverage and more precise guidance in the files generated by the test (namespace App\Tests\) in :

- index.html
- dashboard.html

Good luck and good development !!!!!