



A52 – Qualité de Développement

Romain Orhand
rorhand@unistra.fr

Université

de Strasbourg

IUT Robert Schuman

Institut universitaire de technologie

Université de Strasbourg

De quels éléments du précédent TD vous souvenez-vous ?



```
image: ubuntu

stages:
  - deploy
  - pages

deploy_review:
  stage: deploy
  script:
    - echo "Deploying to review app..."
    - mkdir public
    - cp index.html public/
    - cp script.js public/
  artifacts:
    paths:
      - public
  environment:
    name: review/${CI_PROJECT_NAME}/${CI_COMMIT_REF_NAME}
    url: https://\${CI\_PROJECT\_NAMESPACE}.pages.unistra.fr/-/\${CI\_PROJECT\_NAME}/-/jobs/\${CI\_JOB\_ID}/artifacts/public/index.html
    on_stop: preview_stop
  rules:
    - if: $CI_PIPELINE_SOURCE == "merge_request_event"
```



Éléments de correction du TP

4

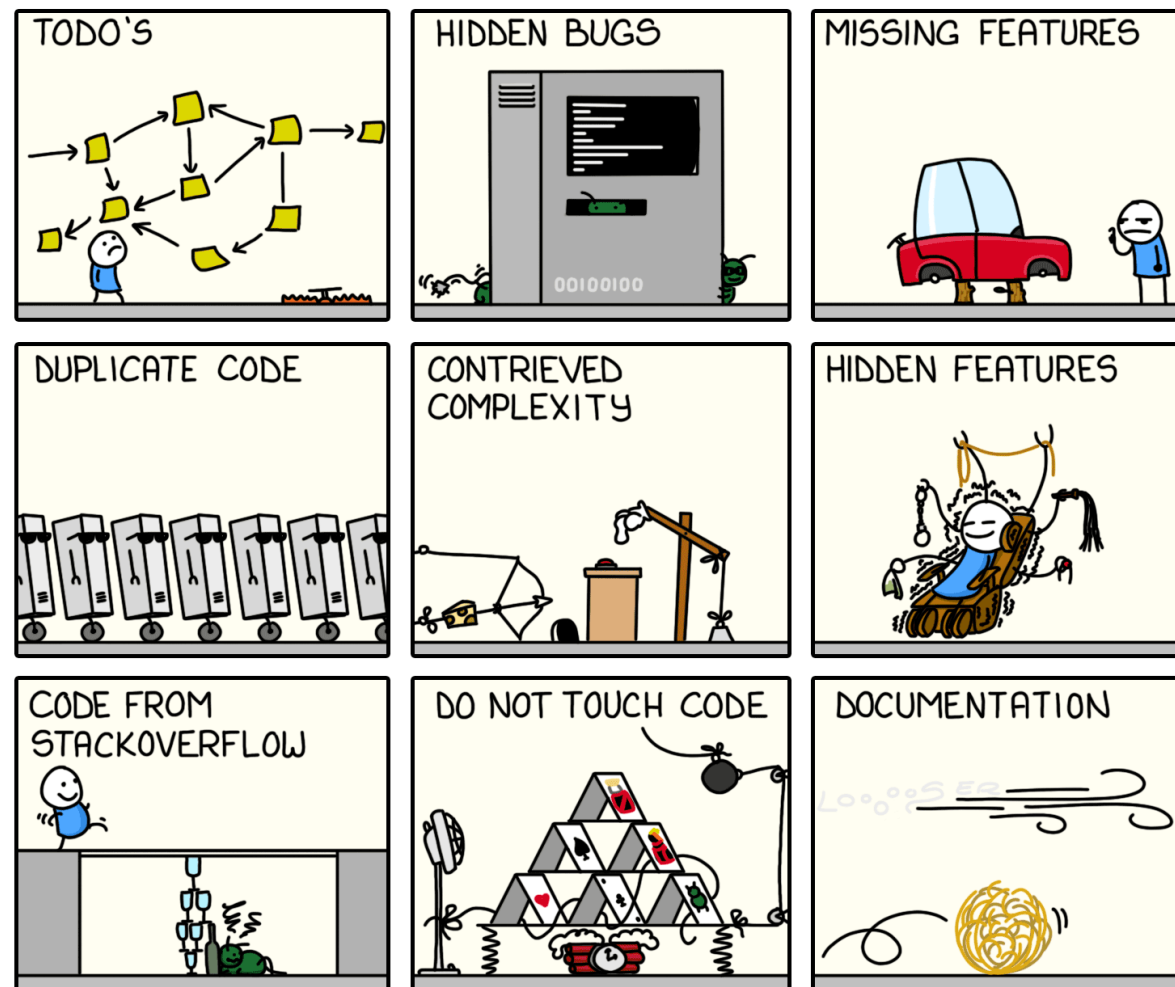
```
preview_stop:
  stage: deploy
  rules:
    - if: $CI_PIPELINE_SOURCE == "merge_request_event"
      when: manual
  allow_failure: true
  environment:
    name: review/${CI_PROJECT_NAME}/${CI_COMMIT_REF_NAME}
    action: stop
  script:
    - echo "Stopping preview"
```



```
pages:
  stage: pages
  script:
    - mkdir .public
    - cp -r * .public
    - mv .public public
  artifacts:
    paths:
      - public
  environment:
    name: production/${CI_PROJECT_NAME}/${CI_COMMIT_REF_NAME}
    url: https://${CI_PROJECT_NAMESPACE}.pages.unistra.fr/${CI_PROJECT_NAME}
  rules:
    - if: $CI_COMMIT_BRANCH == "main"
```

- 1 Documenter
- 2 Outils de documentation
- 3 Exercices
- 4 Projet

POSSIBLE CODE CONTENTS



Quels éléments de documentation connaissez-vous ?

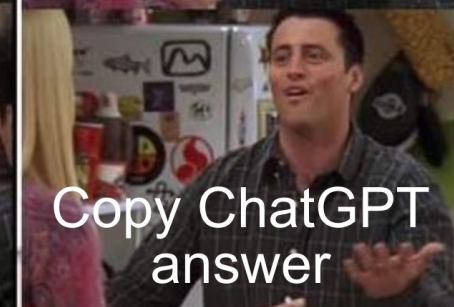
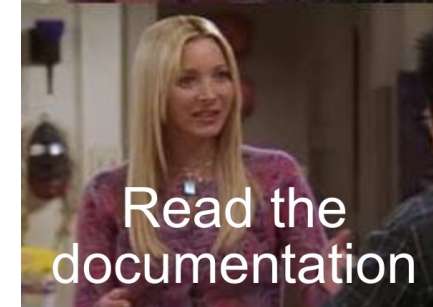
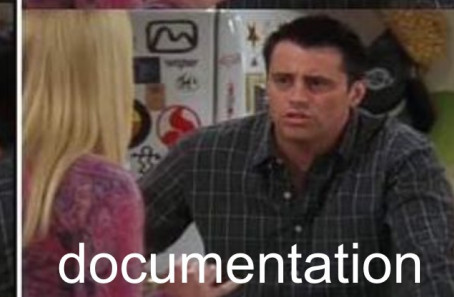
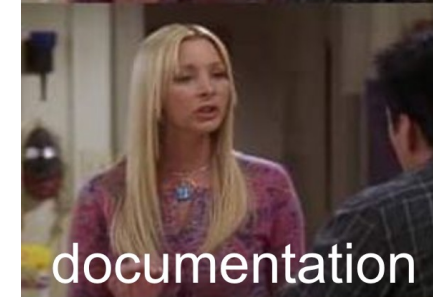
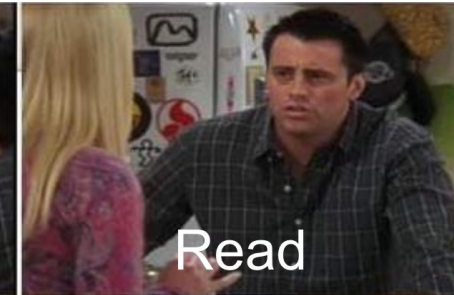
Et ces éléments sont à destination de qui ?

Il s'agit de textes, d'images, de vidéos, ... qui **expliquent** comment un logiciel fonctionne ou comment utiliser un logiciel.

La documentation peut être de plusieurs types et traiter de :

- L'expression de besoin ;
- Architecture / Conception ;
- Technique ;
- Utilisateur ;
- Marketing.

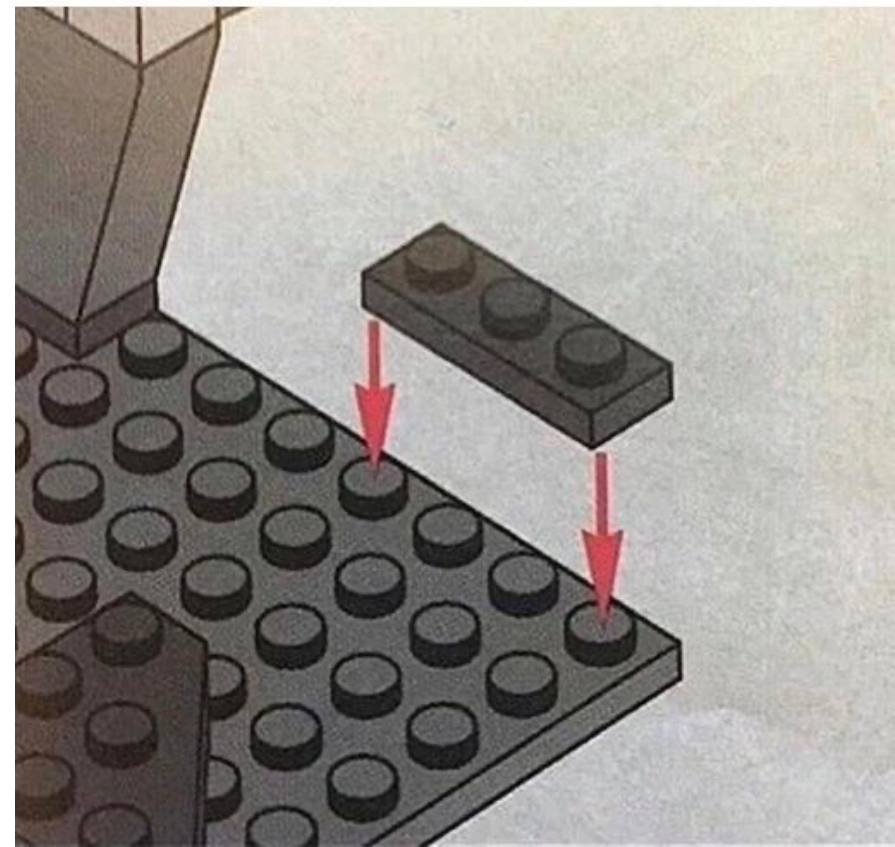
Plus ou moins de documentation ?



1. Il faut **lire**... parfois **beaucoup lire**...
2. La documentation **doit être maintenue à jour** !
3. Il faut **s'adapter** à votre environnement de travail.

Senior Developer: Just read the documentation!

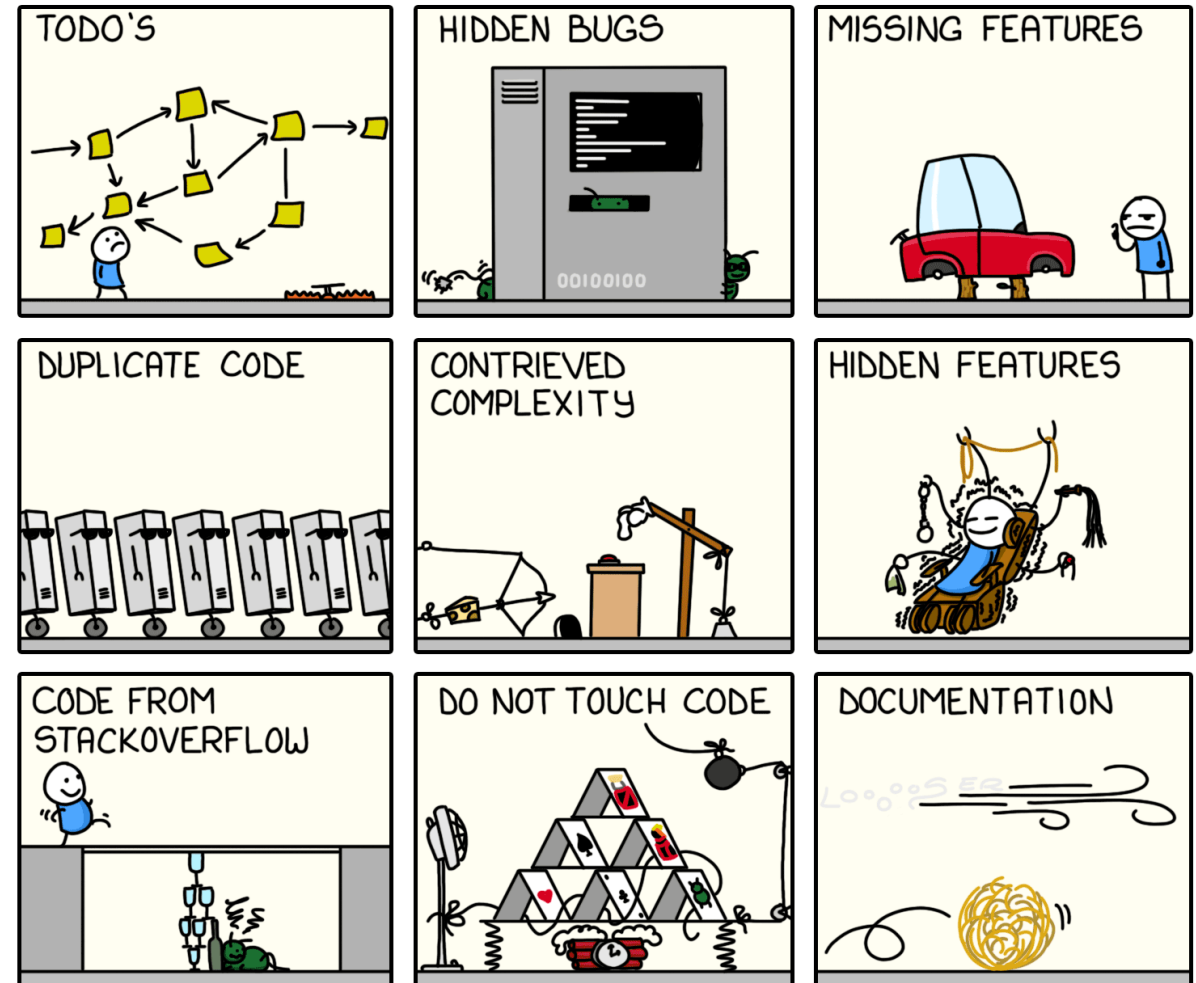
The documentation:



1. Identifier **quand et pourquoi** chaque élément de contexte peut être important.
2. Documenter, brièvement, à **l'endroit exact où** cet élément de contexte aura de l'importance.
3. Veiller à ce que, si cet élément de contexte devient obsolète, il **se modifie, attire immédiatement l'attention sur lui ou disparaisse** complètement.

- 1 Documenter
- 2 Outils de documentation
- 3 Exercices
- 4 Projet

POSSIBLE CODE CONTENTS



Deux extrêmes :

- « *Si votre code a besoin de commentaires, c'est qu'il n'est pas assez clair en soi.* »
- « *Chaque ligne de code devrait être commentée.* »

Quelques conseils :

- Suivez les **conventions** des langages dans lesquels vous programmez et utilisez les **docstrings**.
- S'il n'est **pas possible de fournir un nom explicite** de variable (ou autre), fournissez des commentaires là où vous décrivez votre logique (souvent **les en-têtes, headers, ou docstrings**).
- Expliquez **au niveau des lignes pourquoi** elles ne devraient pas être modifiées si pour une raison ou une autre, elles ne doivent pas être modifiées.

Que rédigez-vous dans un fichier ReadMe ?



Ici, il devient intéressant de documenter **ce qui peut changer**, comme ce que souhaite votre client, le fonctionnement attendu de votre logiciel, etc.



```
class MyFunctionalityShould {  
    @Nested  
    class InSomeContext {  
        @Test  
        void doThisUseCase() {  
            // Arrange - Given  
            // Act - When  
            // Assert - Then  
        }  
        /* Do other tests */  
    }  
    /* Do other tests */  
}
```



On y revient à nouveau
à nos muffins !







<https://www.conventionalcommits.org/fr/v1.0.0/>



**Update .gitignore**
 authored 2 months ago



**Update .gitignore**
 authored 2 months ago

**Update .gitignore**
 authored 2 months ago

**update yml**
 authored 2 months ago

**update yml**
 authored 2 months ago

**update yml**
 authored 2 months ago

**update yml**
 authored 2 months ago

Ces descriptions **sont idéales pour transmettre du contexte à un autre programmeur**, à propos d'un ensemble de modifications opérées sur un code, en particulier **quand il n'est pas à côté de vous**.

Éléments de contexte :

- Comment ce code modifie-t-il l'application ?
- Pourquoi voulons-nous ce changement ?
- Quels sont les compromis associés à ce changement ?
- Si mon *reviewer* débute ou est rouillé dans un langage ou une technologie, quelles informations peuvent l'aider à comprendre mon code, ou comment fonctionne le langage ou la technologie ?

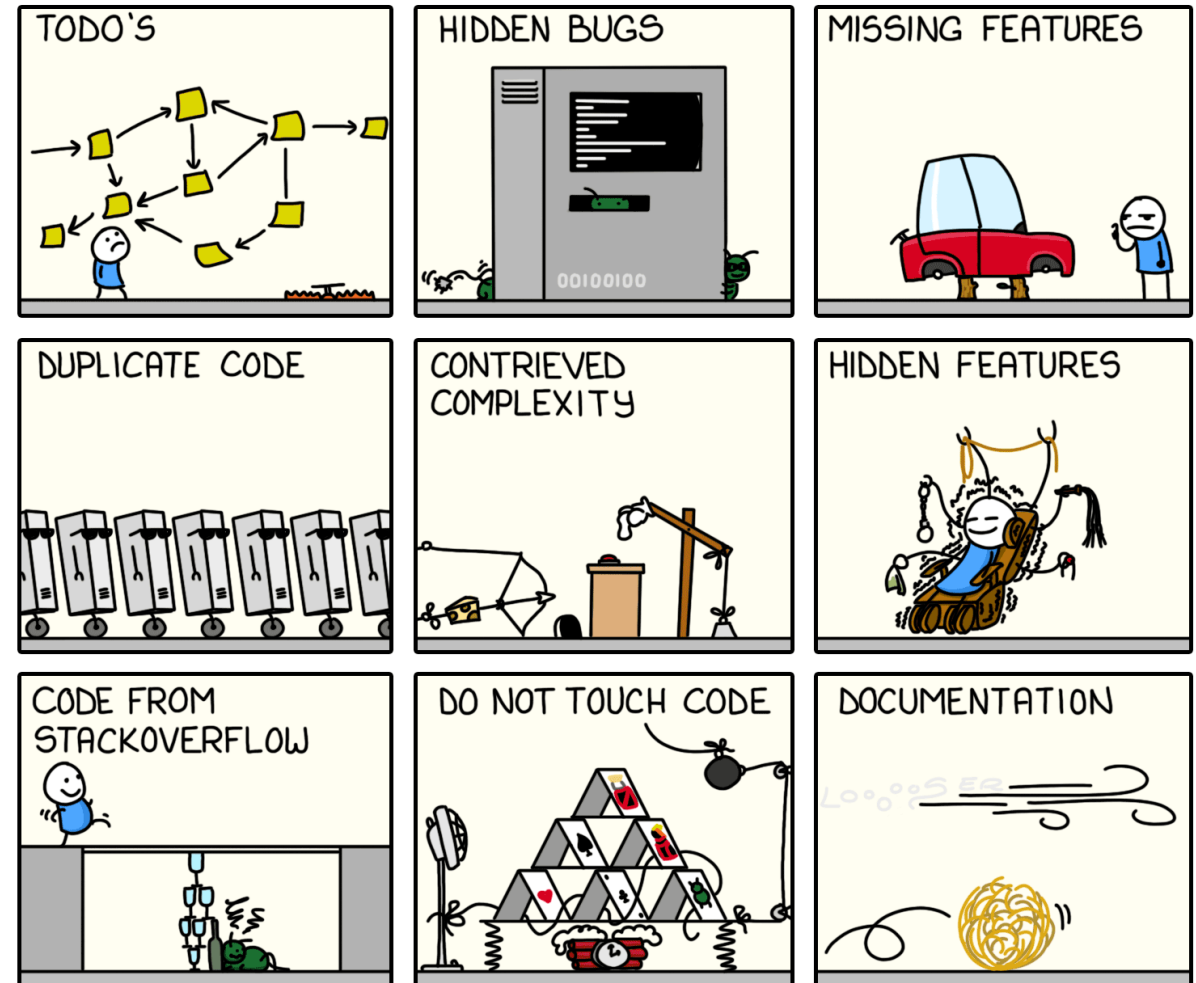
Selon vous,
pourquoi les messages d'erreur peuvent être un outil de documentation pertinent ?

Il existe encore de nombreux autres éléments de documentation !

Petite distinction : certains éléments de documentation peuvent être **automatisés** (certains d'entre vous ont déjà utilisés de tels outils...) et d'autres, pas du tout !

- 1 Documenter
- 2 Outils de documentation
- 3 Exercices
- 4 Projet

POSSIBLE CODE CONTENTS



Retour vers votre [fichier ReadMe](#) issu de projet A51 !

En groupe, pouvez-vous me [commenter](#), à la lumière de ce TD,
[le contenu de votre fichier](#) ?

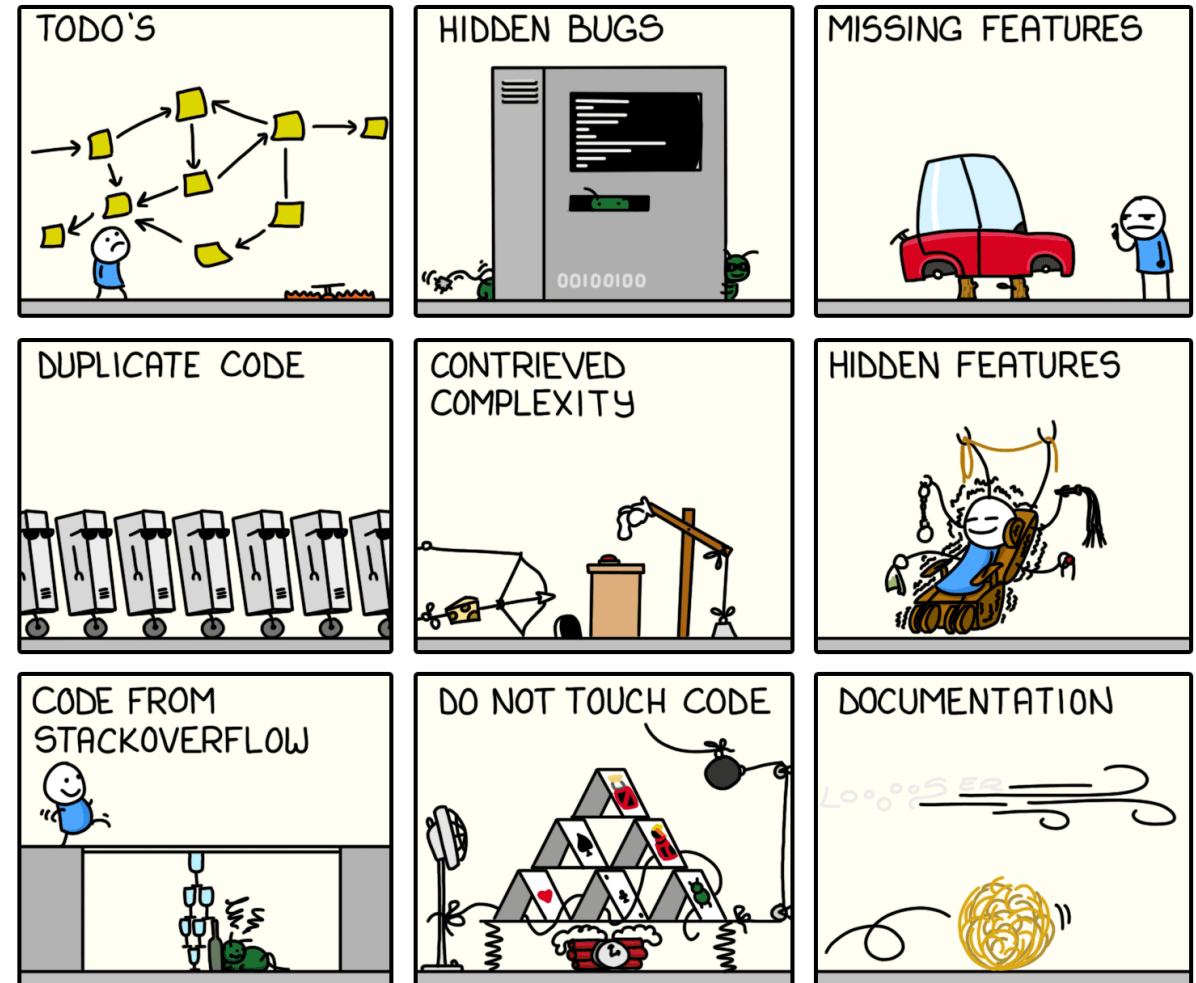
*(Nous savons que vous aviez des consignes particulières à ce propos...
Il n'empêche, que nous proposeriez-vous **maintenant** ?)*

Retour vers votre projet A51, mais vos *commits* cette fois-ci !

Commentez, puis proposez un nouveau message de *commit* si besoin est, pour 5 commits autres que ceux associés à vos *merge requests*.

- 1 Documenter
- 2 Outils de documentation
- 3 Exercices
- 4 Projet

POSSIBLE CODE CONTENTS



Application web cliente à réaliser avec **HTML, CSS et Vanilla JS**.

Mode jeu de rôle.

Rendu sur moodle pour **dimanche 17 décembre 2023 à Midi** :

- ➔ **1 point de pénalité** dès qu'il y a un retard ;
- ➔ Si **le retard est supérieur à 30 minutes**, le rendu ne sera **pas corrigé**.

La partie 2 de ce sujet vous sera transmise via Moodle dans **la soirée de dimanche**.



Pour la prochaine fois

25

Indicateurs de qualité