**Become a Front End Developer**

**with Altimetrik | 2nd Edition**

**Documentation**

[Vanessa Martínez]

Index

***Agile Software Development Methodologies***

*Agile software development, or “Agile”, refers to a group of development methodologies that anticipates the need for flexibility, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams to the delivery of the finished product. It focuses on the clean and fast delivery of individual pieces or parts of the software and not on the entire application.*

*Some benefits:*

*- The ability to help teams in a complex area while it’s still focused on the delivery of business value.*

*- Improves efficiency throughout the organization as teams work together and understand their specific roles in the process.*

*- Companies can trust on who are using agile methodologies because they can feel they are releasing a high-quality product since testing is performed throughout development.*

*- Provide the opportunity to make changes as needed and alert teams to any potential issues*

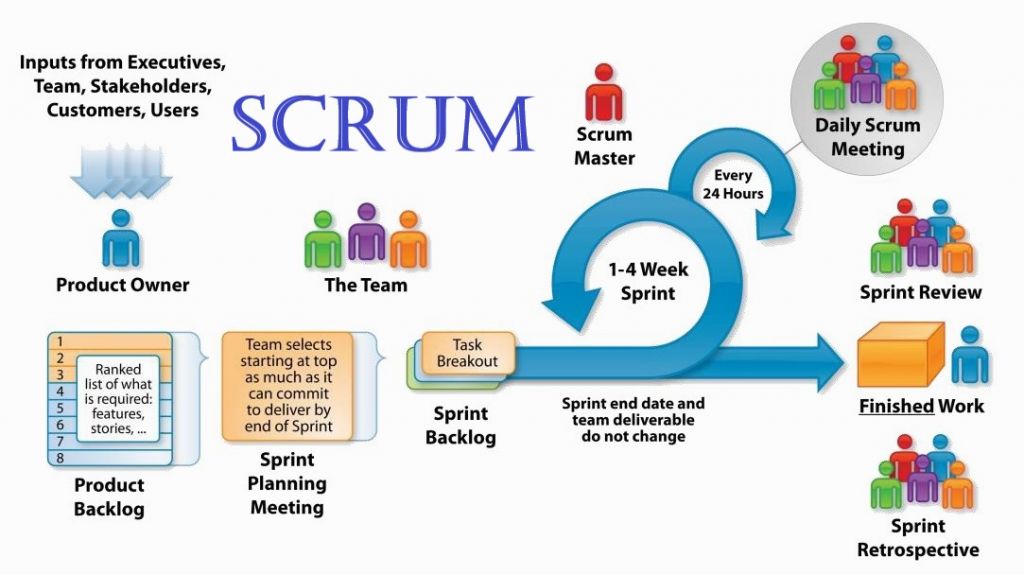
*´.*

***Types of Agile frameworks***

***Scrum***

*Scrum is a lightweight framework, not a methodology, that helps people, teams and organizations to self-organize, and is applied a set of good practices to work together and obtain the best results, generating value by solving complex problems. It replaces a programmed algorithmic approach with a heuristic one (a set of techniques to solve a problem).*

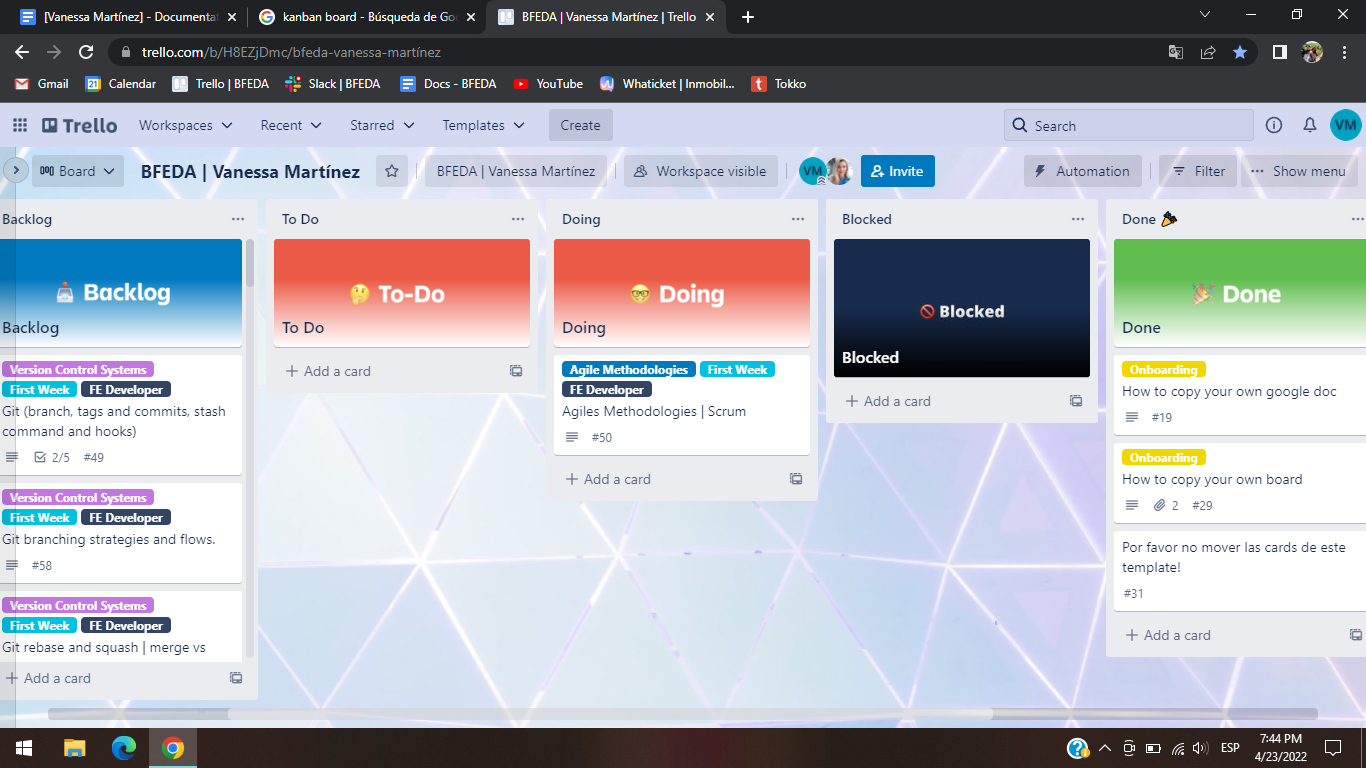
***Roles and process:***

******

***Kanban***

*Kanban is a form to organize tasks with cards on a board that has no limit.*

*You can have a few columns to pass, beginning on the ‘backlog’ column, constantly updated, once the task has begun, you can move the cards to another column called ‘in progress’, or similar, and then through the correspondent columns until it is ‘done’.*

**

***“Scrum is a technique and Kanban is a specific form to apply it”***

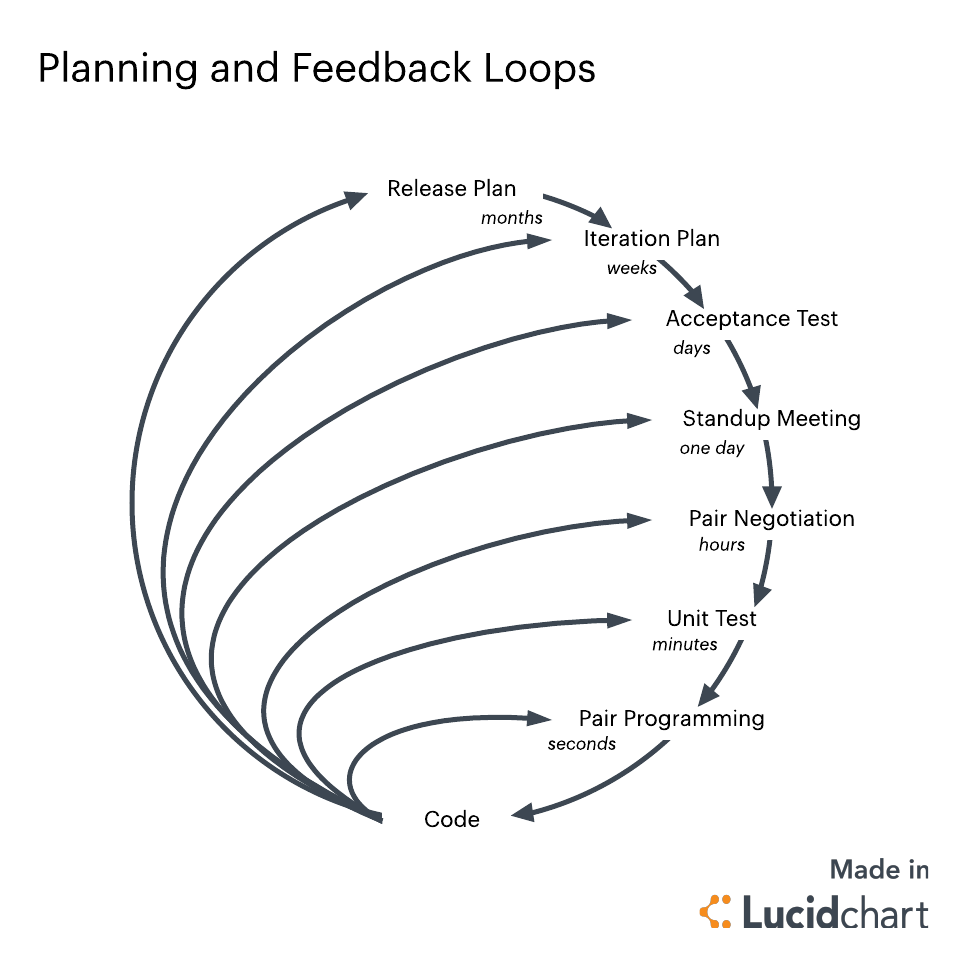
***Extreme Programming (XP)***

*Extreme programming is a framework that aims to produce higher quality software. XP is the most specific of the agile frameworks regarding appropriate engineering practices.*

*XP has 5 values:*

* *Simplicity*
* *Communication*
* *Feedback*
* *Courage*
* *Respect*

*Rules of XP:*

**

***Git***

*Git is a function that controls code’s versions in distributive form that is an important part of daily programming. It allows you to have a complete version history in a fast and free way, this system works with branches and these branches can grow up in different directions from the main one to try out new functionalities.*

***Git commands more common***

* ***Git clone:*** *It’s a command that makes an identical copy of the latest version of a project in your local workspace.*

*git clone <https://name-of-the-repository-link>*

* ***Git branch:*** *It’s used to create, list and delete branches.*

*To create one locally, you use: git branch <branch-name>*

*To push it into the remote repository, you use: git push -u <remote> <branch-name>*

*To list or view it, you use: git branch or git branch --list*

*To delete it, you use: git branch -d <branch-name>*

* ***Git Checkout:*** *It’s mostly used for switching from one branch to another and for checking out files and commits.*

*git checkout <name-of-your-branch>*

*You can also use another command to create and switch a branch at the same time:*

*git checkout -b <name-of-your-branch>*

* ***Git Status:*** *It gives you all the information about branches and files, gathering information like the if the preset branch is up to date, if there is anything to commit, pull or push, if there are files created, modified or deleted.*

*git status*

* ***Git Add:*** *If we create, make a change or erase files, those changes will be on our workspace and won’t be in the next commit. To save it we have to use this command to include those changes into the next commit.*

*To add a file: git add <file>*

*To add everything at once: git add -A*

* ***Git Commit:*** *It’s a command used to save changes we have done after a specific task or issue on our local workspace. We need to write a short message to explain what we did in the source code.*

*git commit -m "commit message"*

* ***Git Push:*** *It sends your changes to the remote server which already are committed.*

*git push <remote> <branch-name>*

*If your branch is new and you need to upload it too, you can use: git push --set-upstream <remote> <name-of-your-branch> or git push -u origin <branch\_name>*

* ***Git Pull:*** *It’s used to get updates from the remote repository and applies the latest changes in your local.*

*git pull <remote>*

* ***Git Revert:*** *It’s used to undo changes in a safe way.*

*First we have to see our commit history, using: git log -- oneline*

*Then specify the hash code of the commit that we want to undo: git revert 3321844*

*Finish pressing shift + q to exit.*

*Really, this command doesn't delete the commit from the history, just create a new “revert” one, you can still see every single move that you did.*

* ***Git Merge:*** *when everything is working ok, it’s used to integrate your feature branch with all of its commits back to the dev branch.*

*Before merging, you have to switch to the dev branch: git checkout dev*

*Then update it using: git fetch*

*At the end, merge your feature branch into dev: git merge <branch-name>*

***Tags and Commits***

*Tags is a feature used to specify a point on a repository. It’s mostly used to keep the release version of a repo, using this tag you are able to move it to an earlier version.*

*git status*

*git add <file>*

*git commit -m “short message”*

*git tag v-1.0 <the correspondent version>*

*git tag -n*

*A* ***tag*** *can be created to a specific* ***commit*** *too, to create it, follow the commands bellow:*

*git status*

*git add <file>*

*git commit -m “short message”*

*git log -- oneline*

***Stash command***

*We can use* ***git stash*** *when we want to temporarily record our current state of the working directory, saving our local modifications away, to manage this copy on another side, here we can make the changes needed without damaging the main and feeling overwhelmed, when everything is fine, you can put it back on the original one.*

***Hooks***

*They are scripts that run every time a certain important action occurs in a repo. Exist two groups of these hooks:*

* ***Client-side hooks:*** *are activated by operations such as committing and merging.*
* ***Server-side hooks:*** *run on network operations such as receiving pushed commits.*

*The most useful local hooks:*

* *pre-commit*
* *prepare-commit-msg*
* *commit-msg*
* *post-commit*
* *post-checkout*
* *pre-rebase*

**Bibliography**

* <https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development>
* <https://www.cprime.com/resources/what-is-agile-what-is-scrum/#:~:text=Agile%20software%20development%20refers%20to%20a%20group%20of%20software%20development,%2Dorganizing%20cross%2Dfunctional%20teams>.
* <https://www.scrum.org/resources/what-is-scrum>
* <https://lucidspark.com/blog/different-types-of-scrum-meetings>
* <https://www.agilealliance.org/glossary/xp/#q=~>(infinite~false~filters~(postType~(~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)
* <https://www.lucidchart.com/blog/what-is-extreme-programming>
* <https://www.freecodecamp.org/news/10-important-git-commands-that-every-developer-should-know/>
* <https://openwebinars.net/blog/que-es-git-y-para-que-sirve/>
* <https://linuxhint.com/add-tag-specific-commit-git/>
* <https://git-scm.com/docs/git-stash>
* <https://www.perforce.com/blog/vcs/git-beyond-basics-stashing-work-progress#:~:text=Using%20the%20git%20stash%20command,more%20easily%20switch%20between%20branches>.
* <https://www.atlassian.com/git/tutorials/git-hooks#:~:text=Git%20hooks%20are%20scripts%20that,in%20the%20development%20life%20cycle>.