TECHNICAL

Glossary

Angular: a JavaScript framework. Allows developers to use HTML as a template language and to extend its syntax.

API: Application **P**rogramming **I**nterface. Specifies how an application's components should interact.

Architecture: a set of structures fundamental in software, their relations, and properties. Also a discipline of planning and designing applications.

Bumping versions: a situation when an update of a **library**/packet/dependency used in a project happens and warrants an upgrade in the project itself. This process can be messy.

Cloud: technology based on physical **servers** and **virtualization**. However, in the cloud, we don't have to worry about hardware. Often linked with **Docker**.

Code review: a practice of ensuring code quality. One developer reads another's code and provides advice/tips on how to create better software.

Deployment: the process of transferring and configuring an application on the desired environment (mainly **servers**).

DevOps: a practice of merging administrator/operations and developer skills. Such a person can develop software and configure servers/cloud infrastructures.

Docker: software for creating containers (small portions of virtual resources: they are different from **VMs**).

DOM: a Document Object Model in a web browser.

DRY: acronym for Don't Repeat Yourself. A principle aimed at reducing the repetition of software patterns.

E2E tests: end-to-end tests. They test an application across all processes.

EcmaScript / ES / JavaScript: programming language.

Framework: a type of software providing architecture solutions and helpers. Simplifies building applications.

GraphQL: a technology developed by Facebook for building **APIs**.

Hermetic code: messily written code. It can quickly become **legacy code**.

Integrations tests: tests showing differences between an app and the app's scope.

JSON: JavaScript Object **N**otation. A file format that uses text readable by humans to transmit data.

KISS: acronym for **K**eep **I**t **S**tupid **S**imple. Implies that most systems work best when kept simple.

Legacy code: old code that is no longer supported (or finding support experts is very hard and expensive).

Library: a portion of code, implementing a portion of a feature.

Native: an application that works without transcompilation (e.g. from **JavaScript** to Kotlin).

NodeJs: technology based on **JavaScript** and working on the client side.

ReactJS: a **JavaScript** library used to create **SPAs**. The library was developed by Facebook and is open-source.

Refactoring: rewriting/cleaning messy code.

Regression tests: tests that show new bugs, or changes in behaviors.

REST: an acronym describing a technology based on HTTP API: **Re**presentational **s**tate **t**ransfer. Provides standards for computer systems that make it easier for them to communicate.

Ruby / Ruby On Rails: Ruby is a programming language, and Ruby On Rails is a **framework** written in Ruby.

Server: a computer dedicated to serving/running applications.

Setup: a process of starting application development (preparing an environment, repository, etc).

SOLID: a mnemonic acronym in object programming that stands for five design principles: **S**ingle responsibility, **O**pen-closed, **L**iskov substitution, **I**nterface segregation, **D**ependency inversion.

SPA: single-**p**age **a**pplication. In a browser, a single-page application fluidly shows new content without the need for reloading (example: Facebook or Gmail).

TypeScript: open-source programming language developed by Microsoft. It transcompiles to **JavaScript**.

Unit tests: granular tests, working on a very low level in code (like functions).

Virtual DOM: a situation when a **DOM** is stored in memory, increasing processing speed.

Virtualization: a term describing techniques leveraged to make a system fully virtual, without linking it to hardware.

VM: a **Virtual Machine**. A system allocated in memory, fully virtual, and separated from hardware.

Vue: a JavaScript framework. Used to build user interfaces on the Web.

YAGNI: acronym for **Y**ou **A**ren't **G**onna **N**eed **I**t. A principle in extreme programming, which implies that no feature should be added until it is absolutely necessary.