**Who is Integration.team?**

Integration.team is a company that specializes in providing solutions for **application integration, business process automation, and enterprise service bus (ESB)**.

We have more than **20 years** of experience in designing, developing, and delivering complex integration projects for various industries and domains.

We use the latest technologies and best practices to ensure high-quality, reliable, and scalable solutions. Some of the tools and platforms we work with include **BizTalk Server, Azure Logic Apps, Nodinite, and Turbo360.**

Our team consists of certified integration experts, architects, developers, and consultants who are passionate about solving integration challenges and helping our clients achieve their business goals.

**We offer a wide range of services, such as:**

* Integration strategy and architecture
* Integration development and testing
* Integration deployment and monitoring
* Integration support and maintenance
* Integration training and mentoring

**Introduction**

**Logging** is the process of recording events and data that occur during a software application’s execution. Logging plays a crucial role in the development cycle of .NET API applications, as it **provides valuable information for debugging, testing, monitoring, and troubleshooting purposes**.   
Logging can help developers to:

* Identify and fix errors in the code.
* Trace the flow and performance of the application.
* Analyze the usage and behavior of the application and its users.
* Audit and verify the security and compliance of the application.
* Generate reports and insights for improvement and optimization.

Logging **can also help stakeholders and clients to understand the value and quality of the application**, and to provide feedback and suggestions for further development. Therefore, logging is an essential practice for creating reliable, robust, and user-friendly .NET API applications.

**About Nodinite**

Nodinite is a platform that enables Microsoft developers to monitor, manage, and troubleshoot their integration applications **in real time**. Nodinite provides a unified dashboard that shows the health, performance, and availability of all your APIs across different environments and platforms.

Nodinite also allows you to drill down into the details of each integration process or API call, such as request and response data, headers, body, errors, logs, and more. Nodinite helps you to identify and resolve issues faster, optimize your integration performance, and ensure high customer satisfaction. With Nodinite, you can **gain insight into your entire API lifecycle**, from development to production, and **improve your API quality and reliability**.

**Internship assignment**

Custom configuration tool for environment management in Nodinite

**Objective:**

Nodinite provides various agents for tools commonly used in integration projects. However, the current system lacks the ability to easily copy configurations across different environments.

**Project Overview:**

Nodinite operates under a single license, allowing users to create multiple environments such as DEV, TST, ACC, and PRD. Often, the same configuration settings need to be replicated across these different environments.

Nodinite’s web client is entirely based on their own API, meaning any action performed in the web client can also be done via API calls. This opens the possibility of developing a configuration tool that simplifies the process of managing settings across environments.

**Scenario:**

For example, consider a scenario where you’ve defined different Log Agents for logging purposes. Each Log Agent is assigned a unique ID. To maintain consistency, you might want to ensure these Log Agents have the same configurations across all environments. Similarly, if you create a log view in one environment, you may want to replicate it in other environments with the same name, access management settings, and other parameters.

These are just two examples of the types of configurations that could be automated across Nodinite environments.

**Your Task:**

You are required to develop a Blazor application that will:

1. Compare configurations across different Nodinite environments.
2. Automate the process of copying configurations from one environment to others using Nodinite's provided APIs.

**Technical Requirements:**

* The Blazor app should be capable of being hosted alongside the existing Nodinite website. Alternatively, it can be hosted in the cloud or within a container, provided it can connect to Nodinite's APIs.

**Outcome:**  
The final product will simplify the management of Nodinite environments by enabling easy comparison and replication of configurations, thereby streamlining the integration process.

**Major Technologies**

* .NET C#
* Blazor
* Azure BICEP
* GitHub Repositories and Actions
* Nodinite + Nodinite API’s