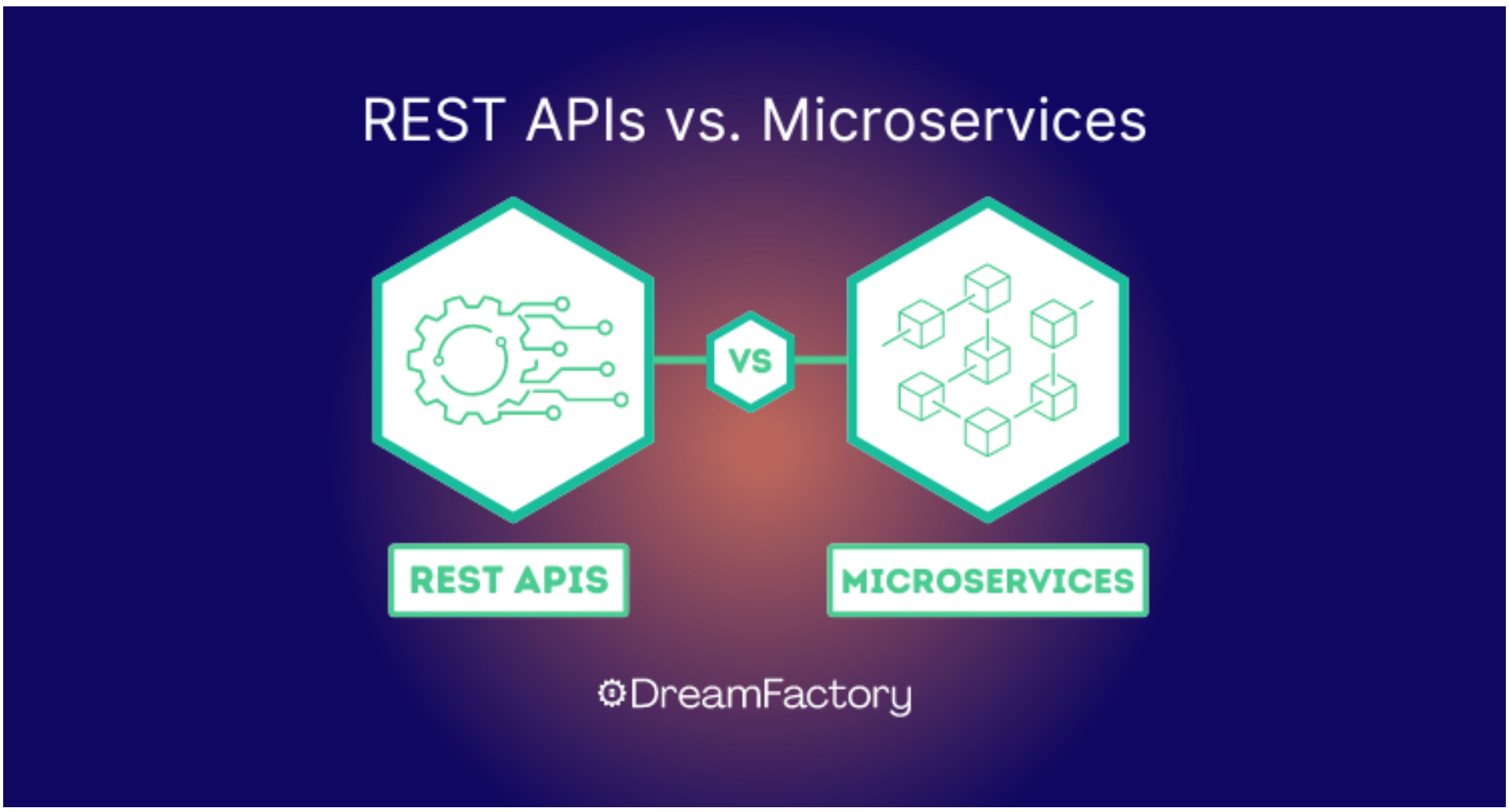
DreamFactory

REST APIs vs Microservices: Key Differences

by Jeremy H • May 20, 2024



The terms RESTful API and Microservices go hand-in-hand when building a microservices-based application. Nevertheless, they refer to very different things.

The easiest way to understand the difference between RESTful APIs and microservices is like this:

- Microservices: The individual services and functions – or building blocks – that form a larger microservices-based application.
- RESTful APIs: The rules, routines, commands, and protocols – or the glue – that integrates the individual microservices, so they function as a single application.

Benefits of Using REST APIs and Microservices

Benefits of Using REST APIs and Microservices

In the world of microservices, REST APIs play a crucial role in facilitating communication between modular components. However, successfully integrating REST APIs with microservices presents several challenges and considerations, which we'll briefly explore in this blog post.

1. API Design and Consistency: Adhering to RESTful API best practices and utilizing common design patterns is essential to maintain uniform interfaces across multiple APIs. Documentation and API specification tools can help achieve consistency.

2. Network Latency and Performance: To minimize latency, optimize API calls by consolidating requests, using pagination, and implementing server-side caching. Technologies like HTTP/2, gRPC, and GraphQL can enhance communication efficiency.

3. Security: Implement authentication and authorization mechanisms such as OAuth 2.0 or JWT to secure data exchange. Using HTTPS encryption and managing API keys diligently further improves security.

4. Error Handling: Follow established conventions for returning status codes and error messages, ensuring clear and actionable information. The Circuit Breaker pattern helps prevent cascading failures between services.

5. Monitoring and Observability: Incorporate logging, tracing, and metrics collection into REST APIs for system insights and issue detection. Tools like Prometheus, Zipkin, and Elasticsearch enable proactive resolution and optimization.

By addressing these key aspects, developers can harness the full potential of REST APIs in microservices-based applications, paving the way for efficient, resilient, and maintainable software solutions.

What is a RESTful API?

Let’s start by defining “API” (application programming interface). An API is a defined set of rules, commands, permissions, or protocols that allow users and applications to interact with – and **Access Data** from – a specific application or microservice. When connecting microservices to create a microservices-based application, APIs define the rules that limit and permit certain actions, interactions, commands, and data-sharing between individual services.

According to **Mahesh Hadlar On Hackernoon**:

The API is an interface, through which many developers interact with the data. A good designed API is always very easy to use and makes the developer’s life very smooth. API is the GUI for developers, if it is confusing or not verbose, then the developer will start finding the alternatives or stop using it. Developers’ experience is the most important metric to measure the quality of the APIs.

One of the most popular types of APIs for building microservices applications is known as “RESTful API” or “REST API.” REST API is a popular standard among developers because it uses HTTP commands, which most developers are familiar with and have an easy time using. Here are the defining characteristics of RESTful API:

- An API that uses the REST (representational state transfer) model.
- Relies on HTTP coding which is familiar to web developers.
- Uses SSL (Secure Sockets Layer) encryption.
- Language agnostic in that you can use to connect apps and microservices written in different programming languages.
- REST APIs allow you to create a web application with CRUD operations (create, retrieve, update, delete).

The commands – or “verbs” – common to REST API include:

- HTTP PUT
- HTTP POST
- HTTP DELETE
- HTTP GET

- HTTP PATCH

Developers use these RESTful API commands to perform actions on different “resources” within an

application or service. Those resources could be data in a database that pertains to employees, accounting details, medical records, or many other things. In addition to resources, RESTful APIs use URLs (Uniform Resource Locators), which allow you to locate and indicate the resource you want to perform an action on.

The familiarity and reliability of RESTful API commands, rules, and protocols make it easier to develop applications that integrate with applications that have an associated API. This is especially true when a company like Instagram, Twitter, Salesforce, or Facebook want to make their services available to integrate with third-party applications.

The Facebook Messenger Platform is an excellent example of RESTful APIs in action. Messenger is comprised of a variety of microservices – such as “Send” for sending messages and “Attachment” for attaching and sending files. To help third-party developers create applications that integrate with these and other messenger services, Facebook has published several RESTful APIs.

Facebook describes its **REST APIs For Messenger** like this:

The Messenger Platform provides a set of REST APIs that give you the tools you need to create awesome Messenger experiences. From sending rich messages, to finding your existing customers on Messenger, to customizing your bot and more, our APIs are the primary way you will work with the Messenger Platform.

Here are three REST APIs available for the Messenger Platform:

- Send API: Serves as the primary way for apps to integrate with the Messenger Platform. The Send API allows developers to create third-party applications that log into an account on Facebook Messenger and send and receive text messages and file attachments. This allows developers to build auto-response bots, chatbots, and other services that integrate with the Messenger platform.
- Attachment Upload API: Allows your app to upload and send images, audio, videos, and files through messenger. This API also makes uploaded assets reusable, so you don’t have to keep uploading files that you send repeatedly.
- Messaging Insights API: Allows your app to retrieve Page Insights related to your Facebook page. This makes different Facebook metrics available to your app – particularly those related to conversation numbers, responsiveness, block rate, and more.

RESTful APIs for Facebook, Salesforce, and countless other cloud-based applications and microservices have empowered app developers to connect and integrate their applications with the widest array of platforms and services. This gives platform developers greater power, flexibility, extensibility, and reach – and it encourages innovation in ways the original platform developers may not have imagined.

What Are Microservices?

The best way to explain microservices – and microservices-based applications – is to start with a description of monolithic applications.

Enterprises have long relied on monolithic applications to run their operations and provide different functions and services to their customers. All the code for a monolithic application’s services and functions is found within the same piece of programming.

TechTarget describes monolithic applications like this:

Monolithic software is designed to be self-contained; components of the program are interconnected and interdependent rather than loosely coupled as is the case with modular software programs. In a tightly-coupled architecture, each component and its associated components must be present in order for code to be executed or compiled.

Furthermore, if any program component must be updated, the whole application has to be rewritten, whereas in a modular application, any separate module (such as a microservice) can be changed without affecting other parts of

the program.

A monolithic application architecture makes sense when an enterprise is first starting out, but eventually,

enterprises will need to upgrade and scale their monoliths as their businesses and customer-bases grow. This can be challenging because the code for the monolith's services and functions is tightly-coupled and highly interdependent.

The more developers bolt new services and functions onto the monolith, the more difficult it becomes to untangle the code for future upgrades. Eventually, it's virtually impossible to change even a small part of the monolith without **Refactoring** the entire application. Moreover, scaling a monolithic application is also inefficient because developers have to scale the whole application – instead of simply scaling an individual function or service.

The upgrading and scaling challenges of monolithic applications eventually led developers to create the microservices-based application architecture. The microservices architectural style breaks the monolith into its component functions and services. Then it develops and runs each service as a small, autonomous, independent application, i.e., microservice. Finally, it loosely connects these microservices – usually with RESTful APIs – so they work together to form the larger application.

Techopedia describes microservices like this:

Microservices is the idea of offering a broader platform, application or service as a collection of combined services. These microservices provide specialized, fine-grained cooperation that makes up the more comprehensive architecture model.

The use of microservices in apps can be structured in many different ways. Within the application, a microservice does one defined job – for example, authenticating users, generating a particular data model or creating a particular report. The idea is that these microservices, which are often language-agnostic, can fit into any type of app and communicate or cooperate with each other to achieve the overall goal.

APIs vs Microservices: How They Work Together

Now that you have a better understanding of RESTful APIs and microservices, you can see how these two concepts work together to build a microservices-based application architecture: Microservices function as the "building-blocks" of the application by performing various services, while "RESTful APIs" function as the "glue" that integrates the microservices into an application.

When developers use RESTful APIs and microservices to create a modular, service-oriented architecture like this, enterprises can achieve the following benefits:

- Scalability: Applications are easier and more cost-effective to scale because you only need to scale the services that need it and not the entire architecture.
- Cost savings: Cost savings on development since upgrades are faster and easier.
- Resiliency: Greater ability to prevent and contain failure cascades.
- Easy upgrades: A pluggable, modular architecture that facilitates adding, upgrading, or removing services and functions faster and with less chance of coding conflicts.
- Rapid development: Faster time to market when it comes to developing new functions and services.
- Security and compliance: Improved data security and compliance due to the containment of microservices – which work independently and don't have knowledge of each other. RESTful API connections between microservices let you define strict rules for data access and sharing.
- Language agnostic: The ability to connect modular services that were programmed in different languages regardless of what kind of platform they're running on.
- Smaller teams: Smaller, more agile development teams for each microservice.
- Agility: Greater agility to respond faster to changing business needs.
- Cloud-based architecture: Microservices usually run on cloud-based services like Amazon AWS.

For a more detailed discussion of these benefits, please read our guide on the "**Benefits Of Microservices**."

Challenges and Considerations when using REST APIs with Microservices

The rise of microservices architecture has revolutionized the way modern software applications are designed and developed. With their focus on modularity, flexibility, and scalability, microservices have become the go-to approach for many developers. One popular way to implement communication between these modular services is using REST APIs. While the combination of REST APIs and microservices can lead to highly efficient and maintainable applications, there are several challenges and considerations to keep in mind. In this section, we will explore the key aspects that developers must address to ensure the successful integration of REST APIs with microservices.

API Design and Consistency:

A well-designed API is crucial for the seamless interaction between microservices. However, ensuring consistency across multiple APIs can be challenging. To maintain a uniform interface, developers should adhere to a standardized set of principles, such as **RESTful API Best Practices**, and utilize common design patterns. Additionally, documenting the API design and using API specification tools like Swagger or OpenAPI can help ensure a consistent and maintainable API ecosystem.

Network Latency and Performance:

The distributed nature of microservices often results in increased network latency, which can negatively impact application performance. To minimize latency, developers should optimize API calls by consolidating multiple requests into a single one, using pagination, and implementing server-side caching. Employing technologies such as HTTP/2, gRPC, or GraphQL can also help improve communication efficiency between microservices.

Security

Securing communication between microservices is critical to protect sensitive data and prevent unauthorized access. Developers should implement authentication and authorization mechanisms, such as OAuth 2.0 or JSON Web Tokens (JWT), to ensure secure data exchange between services. Additionally, using HTTPS encryption and carefully managing API keys can further enhance the security of REST APIs.

Error Handling

Robust error handling is essential to ensure the reliability and resiliency of applications built with microservices. When designing REST APIs, developers should follow established conventions for returning status codes and error messages, providing clients with clear and actionable information. Implementing the **Circuit Breaker Pattern** can also help prevent cascading failures between services.

Monitoring and Observability

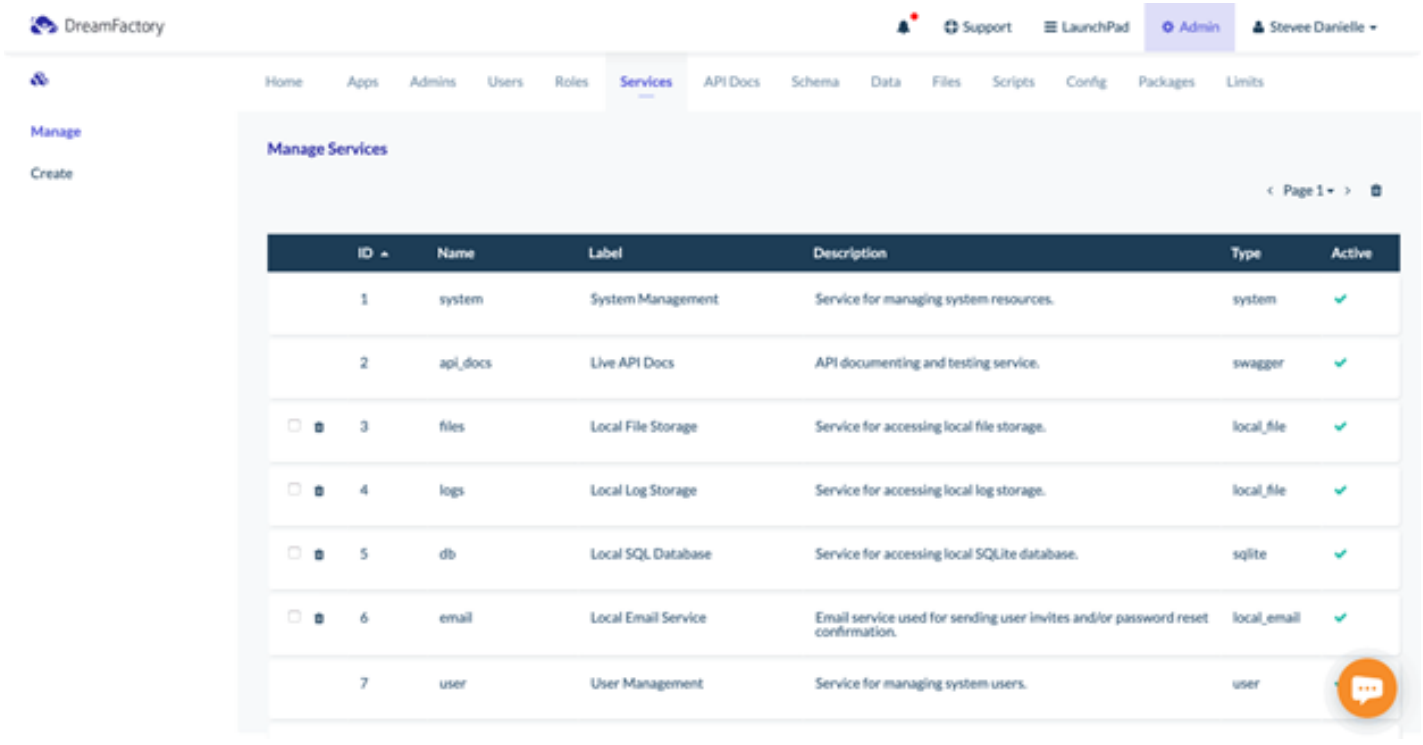
Monitoring and observability are critical for maintaining the health and performance of microservices-based applications. Developers should incorporate logging, tracing, and metrics collection into their REST APIs to gain insights into system behavior and detect potential issues. Tools such as Prometheus, Zipkin, and Elasticsearch can be employed to aggregate and analyze this data, facilitating proactive issue resolution and system optimization.

How to Build RESTful APIs in Minutes

Having discussed the differences between RESTful APIs and microservices, there's one issue we haven't addressed: The fact that it takes a tremendous amount of time to hand-code custom RESTful APIs for each microservice in the architecture. It can take a developer three-weeks to hand-code a simple RESTful API.

You can bypass this development time with a modern iPaaS (Integration Platform as a Service) like DreamFactory. The **DreamFactory IPaaS** includes an automatic API generation feature that helps you convert any database into REST API in just a few minutes. DreamFactory also allows you to instantly convert a SOAP Web Service into REST API – which applies REST endpoints to your SOAP APIs, which makes them easier to work with.

Here’s a snapshot of DreamFactory's RESTful API generation:



Schedule A Free, Hosted Trial Of The DreamFactory Platform Now!

Conclusion

Successfully implementing REST APIs with microservices requires careful consideration of various challenges and best practices. By addressing API design and consistency, network latency and performance, security, error handling, and monitoring and observability, developers can create robust, efficient, and scalable applications that leverage the full potential of microservices architecture. As the technological landscape continues to evolve, it is essential for developers to stay informed about emerging trends and advancements that can further optimize their REST API and microservices-based solutions. By doing so, they will be well-equipped to deliver high-quality software that meets the dynamic needs of modern businesses and users, setting the stage for continued success in the ever-competitive digital world.

Frequently Asked Questions: REST APIs vs Microservices

What are REST APIs?

REST (Representational State Transfer) APIs are architectural principles used for designing networked applications. They provide a set of guidelines and constraints to build scalable, stateless, and interoperable web services.

What are Microservices?

Microservices refer to an architectural style where applications are decomposed into small, independent services that are loosely coupled and communicate with each other via APIs. Each microservice handles a specific business capability.

How do REST APIs and Microservices relate to each other?

REST APIs are commonly used as the communication mechanism between different microservices in a

Microservices architecture. Microservices expose their functionalities as RESTful APIs, allowing other microservices or external systems to interact with them.

What are the key differences between REST APIs and Microservices?

1. REST APIs focus on the communication protocol and design principles for building web services, while Microservices refer to an architectural style for building applications.

2. REST APIs can be used within monolithic applications as well, but Microservices architecture specifically emphasizes the decomposition of applications into smaller services.

3. REST APIs are a communication mechanism, while Microservices encompass a broader architectural approach involving service independence, scalability, and fault isolation.

Can you have Microservices without REST APIs?

Yes, while RESTful APIs are commonly used in Microservices architectures, other communication protocols can also be employed, such as messaging queues or event-driven mechanisms. REST APIs are not a mandatory requirement for implementing Microservices.

What are the advantages of using REST APIs?

REST APIs offer a standardized approach to designing and exposing web services, promoting scalability, ease of integration, and platform independence. They enable client-server communication over the HTTP protocol, allowing various clients (web, mobile, IoT) to interact with the services.

What are the benefits of adopting a Microservices architecture?

Microservices Architecture provides benefits such as improved scalability, flexibility, independent service development and deployment, fault isolation, technology diversity, and increased team autonomy. It enables faster development cycles, supports continuous deployment, and allows teams to work on different services simultaneously.

When should I consider using REST APIs within a Microservices architecture?

REST APIs are typically used for inter-service communication within a Microservices architecture. They provide a standard, language-agnostic way for microservices to interact and exchange data. RESTful principles can be employed to design the API contracts between services.

Are there any challenges associated with using REST APIs and Microservices?

Yes, challenges may include managing service dependencies, ensuring proper versioning and backward compatibility of APIs, implementing effective API documentation, handling distributed Data Consistency, and addressing complexities in service orchestration and monitoring.

Which is better, REST APIs or Microservices?

REST APIs and Microservices are not mutually exclusive choices. REST APIs are a communication mechanism, whereas Microservices represent an architectural style. REST APIs are commonly used within Microservices architectures. The choice between the two depends on the specific needs, requirements, and context of your application.

Related Reading

What Is A Microapp: An Emerging Trend

The microapp trend is on the rise! In the approximately two years since joining the DreamFactory team, I'd estimate I've conversed with more than one thousand companies about their API-based projects. These conversations provide a great opportunity to peer inside the IT operations of organizations ... Continue Reading

DreamFactory Software - Blog

Jeremy H

Fascinated by emerging technologies, Jeremy Hillpot uses his backgrounds in legal writing and technology to provide a unique perspective on a vast array of topics including enterprise technology, SQL, data science, SaaS applications, investment fraud, and the law.

RECOMMENDED ARTICLES

A Complete Guide to API Generation

10 Best API Management Tools

Creating a Microsoft SQL Server API in Less Than 5 minutes with DreamFactory

Hasura vs. DreamFactory: A Comprehensive Comparison

Build A Snowflake REST API in Less Than 5 Minutes

DreamFactory

info@dreamfactory.com

sales@dreamfactory.com

+1 415-993-5877

API Generation

Features

Connectors

Snowflake

Developers

Docs

Guide

Github

API Calculator

Company

About Us

Contact Us

Support