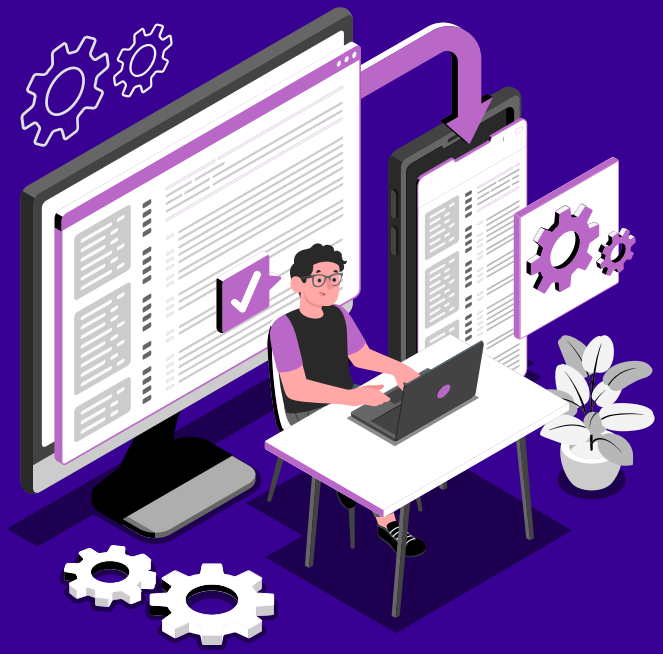

API (Application Programming Interface)

Présenté par:
Noura Abdenaim
Bachir cherif youssef

2022/2023

Plan de travail

01 Qu'est-ce qu'une API ?

02 Qui a créé l'AP ! & utilisation de l'AP

03 Types d'API

04 Implimentation de API

Qu'est-ce qu'une API ?

Les API sont des mécanismes qui permettent à deux composants logiciels de communiquer entre eux à l'aide d'un ensemble de définitions et de protocoles. Par exemple, le système logiciel du bureau météorologique contient des données météorologiques quotidiennes. L'application météo de votre téléphone "parle" à ce système via des API et vous montre des mises à jour météo quotidiennes sur votre téléphone.

Une API (application programming interface ou « interface de programmation d'application ») est une interface logicielle qui permet de « connecter » un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités.

Qui a créé l'API ?

L'API a été développé au départ par des professeurs de langue britanniques et français sous la direction de Paul Passy dans le cadre de l'Association phonétique internationale, fondée à Paris en 1886 sous le nom de Dhi Fonètik Tîcerz' Asóciécon.

Pourquoi l'API est-elle utilisée ?

Les API sont nécessaires pour rassembler les applications afin d'exécuter une fonction conçue autour du partage de données et de l'exécution de processus prédéfinis. Ils fonctionnent comme un intermédiaire, permettant aux développeurs de créer de nouvelles interactions programmatiques entre les différentes applications que les gens et les entreprises utilisent quotidiennement.

Types d'API et comment choisir lequel construire

Les interfaces de programmation d'application, mieux connues sous leur petit nom d'API, libèrent les données pour permettre aux entreprises de connecter systèmes, applications, appareils et ensembles de données. Pour savoir quel type d'API convient le mieux à tel ou tel projet, il faut connaître le cas d'utilisation envisagé, les personnes qui utiliseront ces API ou y accèderont, ainsi que les systèmes et ensembles de données à connecter. Pour que les performances et la gestion des API soient optimales, il faut déterminer quel type d'API convient le mieux afin de concevoir et de construire l'architecture en conséquence.

Types d'API par utilisation

Les API système : les API système libèrent les données des systèmes d'enregistrement stratégiques de l'entreprise.

Les API de processus : les API de processus interagissent avec les données et les façonnent au sein d'un système unique ou sur plusieurs systèmes, ce qui a l'avantage d'éliminer les silos.

Les API d'expérience : les API d'expérience fournissent un contexte aux données et processus libérés et établis à l'aide d'API système et d'API de processus.

Types de stratégies de gestion des API

API externes :
Les API externes sont accessibles par des tiers (développeurs, partenaires, etc.) extérieurs à l'entreprise. Elles facilitent généralement l'accessibilité des données et des services de l'organisation sous forme de libre-service destiné aux développeurs du monde entier qui souhaiteraient créer des applications et des intégrations innovantes
API internes :
Les API internes sont diamétralement opposées aux API ouvertes. En effet, les utilisateurs externes ne peuvent pas y accéder, et elles sont uniquement mises à disposition des développeurs en interne.
API partenaires:
Les API partenaires se situent à mi-chemin entre les API internes et externes. Elles sont accessibles par des utilisateurs extérieurs à l'entreprise disposant d'autorisations exclusives. D'ordinaire, cet accès spécial est accordé à des tiers bien spécifiques dans le but de faciliter un partenariat métier stratégique.

Types de style d'architecture d'API

REST : REST (Representational State Transfer) est un style d'architecture qui sépare les préoccupations de l'utilisateur d'API du fournisseur d'API grâce à des commandes intégrées au protocole de réseau sous-jacent. Les clients se servent des liens et des formulaires inclus pour exécuter les actions (par exemple, la lecture, la mise à jour, le partage, l'approbation, etc.). Le HTML est l'exemple par excellence de ce style, mais il existe de nombreux formats dédiés aux API (HAL, CollectionJSON, Siren, etc.).

RPC : dans le cas des RPC ou Remote Procedure Calls, les développeurs doivent généralement exécuter des blocs spécifiques de code sur un autre système. Les appels de procédure à distance de type RPC sur d'autres systèmes nécessitent généralement l'intervention de développeurs qui doivent nommer ces procédures. RPC est indépendant, ce qui signifie qu'il est pris en charge par de nombreux protocoles, sans toutefois bénéficier des avantages offerts par l'utilisation des fonctionnalités des protocoles natifs (par exemple, la mise en cache).

Orientée événements/de streaming : les API orientées événements, que l'on nomme aussi architectures événementielles, en temps réel, de streaming, asynchrones ou encore push, n'attendent pas qu'un utilisateur d'API fasse appel à elles pour donner une réponse. Au lieu de cela, c'est la survenue d'un événement qui déclenche la réponse

Implémenter de l'API

1. Déterminez l'objectif de votre API et définissez ses fonctionnalités. Quel type de données fournira-t-il ou acceptera-t-il ? Comment les utilisateurs y accéderont-ils ?

2. Choisissez le langage de programmation, les frameworks et les outils que vous utiliserez pour créer votre API. Les choix populaires incluent Node.js, Python, Ruby on Rails et Flask.

3. Décidez du type d'API que vous souhaitez créer. S'agira-t-il d'une API RESTful, d'une API SOAP ou d'une API GraphQL ?

4. Créez une conception et une architecture détaillées pour votre API. Cela comprendra la création d'un modèle de données, la définition des points de terminaison et la détermination de la manière dont l'API gérera l'authentification et la gestion des erreurs.

5. Commencez à coder votre API. Utilisez les outils et les infrastructures que vous avez choisis pour créer les points de terminaison, ajouter une logique pour gérer les demandes et définir les formats de réponse.

6. Testez soigneusement votre API. Utilisez des outils de test automatisés comme Postman ou curl pour vous assurer que votre API fonctionne comme prévu.

7. Documentez votre API. Créez une documentation claire et concise qui explique comment utiliser votre API, y compris les points de terminaison, les paramètres et les réponses attendues.

8. Déployez votre API. Choisissez un fournisseur d'hébergement qui peut gérer le trafic que votre API recevra et déployez votre API auprès de ce fournisseur.

9. Surveillez et maintenez votre API. Surveillez en permanence les performances et l'utilisation de votre API, et apportez des améliorations et des mises à jour si nécessaire.

Exmple

Dans cet exemple, nous définissons un modèle de données pour une ressource utilisateur et créons des points de terminaison pour obtenir une liste d'utilisateurs, obtenir un utilisateur spécifique par ID, créer un nouvel utilisateur, mettre à jour un utilisateur existant et supprimer un utilisateur. Nous utilisons également des codes d'état HTTP pour indiquer le succès ou l'échec des demandes et fournissons des réponses JSON pour l'échange de données.

Pour exécuter cet exemple d'implémentation d'une API RESTful dans Node.js à l'aide du framework Express, procédez comme suit :
1. Installez Node.js et npm .

2. Créez un nouveau répertoire pour votre projet et accédez-y dans votre terminal ou votre invite de commande.

3. Initialisez un nouveau projet Node.js en exécutant "npm init" et en suivant les invites. Cela créera un fichier package.json dans votre répertoire de projet.

4. Installez le package Express en exécutant "npm install express".

5. Créez un nouveau fichier appelé `server.js` (ou quel que soit le nom que vous souhaitez lui donner) et copiez-y l'exemple de code d'implémentation.

6. Exécutez le serveur en exécutant `node server.js` dans votre terminal ou invite de commande. Cela démarrera le serveur sur le port 3000 par défaut.

7. Utilisez un outil tel que Postman ou cURL pour tester les points de terminaison de votre API en envoyant des requêtes HTTP à `http://localhost:3000` (ou tout autre port que vous avez spécifié dans la variable `PORT`).

Conclusion

En conclusion, les API sont un composant essentiel du développement logiciel moderne, permettant à différentes applications de communiquer et d'échanger des données de manière standardisée et efficace. Les API ont révolutionné la façon dont nous créons et connectons des applications logicielles, permettant aux développeurs d'accéder plus facilement à des services et à des sources de données puissants et de créer des applications plus interconnectées et plus puissantes. À mesure que la technologie continue d'évoluer, les API joueront probablement un rôle encore plus important dans le développement de logiciels, permettant de nouveaux niveaux d'intégration et de fonctionnalité sur différentes plateformes et services.