World Journal of Educational Research
ISSN 2375-9771 (Print) ISSN 2333-5998 (Online)
Vol. 11, No. 2, 2024
www.scholink.org/ojs/index.php/wjer

Original Paper

Leveraging Cloud Infrastructure for Rapid Application Development and Deployment of Cloud Services

Paul M. Okanda[1]

[1] Computing Department, School of Science and Technology, United States International University-Africa, Nairobi, Kenya

Received: February 11, 2024 Accepted: March 11, 2024 Online Published: April 18, 2024
doi:10.22158/wjer.v11n2p166 URL: http://dx.doi.org/10.22158/wjer.v11n2p166

Abstract

Backend as a service (BaaS) is service model in which developers outsource all the behind-the-scenes aspects of a web or mobile application so that they only have to write and maintain only the frontend. This paper explores representative BaaS platforms and the limitations they present to users. Using the feedback received from the developer community, the author presents how the research project addresses the weaknesses earlier identified in existing BaaS platform. Some of these concerns include security aspects that a BaaS platform should present as a bare minimum. Finally, the paper discusses a summary of the benefits that businesses and developers employing a BaaS platform could derive from using a BaaS platform that addresses the limitations of existing platforms. The platform, referred to as DbAPI symbolizes the ways in which databases and Application Programming Interfaces (APIs) present a powerful tool that provides cloud services to users in a way that abstracts the complexity of deploying cloud services while still addressing security concerns. This tool enhances the software development process through the acceleration of the software development life cycle, the reduction of associated costs, and the abstraction of computing infrastructure complexity.

Keywords

Cloud, Backend as a Service, Databases, APIs, Infrastructure Automation

1. Introduction

Cloud refers to servers accessible via the internet, along with the software and databases operating on these servers (CloudFlare, 2023). One key technology behind cloud computing is known as virtualization. This enables the development of a simulated, digital-only "virtual" computer that mimics the behavior of a physical computer, complete with its own hardware.

There are numerous BaaS platforms that are widely used by developers. Two of the most widely used platforms are presented below. Their implementation details, the benefits they offer, their major limitations which form the basis for the objectives of the new BaaS platform called DbAPI are discussed in an effort by the author to address these concerns.

Firebase is a platform backed by Google and is described as ‘an app development platform that helps you build and grow apps and games’ (Firebasem n.d.) on their website, The researchers categorize it as a BaaS platform because the services it offers developers, accurately fit the definition above. Firebase, as described on their website offers products related to building applications fast, securely and at scale (Firebase, n.d.). Firebase offers cloud databases, authentication, and storage among other things. These three are listed because they are the core of any backend service. Firebase provided access to these services to create a backend for applications without really diving deep into programming how they work, effectively providing a backend-as-a-service platform.

While exploring Firebase, the researchers discovered that putting immense effort towards designing a backend on Firebase is very imperative. In other words, developers must implicitly design how they want the backend to function. Although, there are many tools that abstract features, if one is developing a big software system, the customization of Firebase features takes on even more significance. This means the developer will to some reasonable extent need to learn how to use Firebase as an additional technology, in much the same way they would learn a programming language as opposed to simply utilizing a platform that provides required services and learning it as a tool.

Furthermore, when programming a frontend that relies on the Firebase BaaS, the developer has to use specialized libraries that the platform offers. This is an additional learning curve which has a similar experience to acquiring skills in a new language. Should the project require substantive effort, this is comparable to doing backend programming on the frontend. The author’s considered opinion is that a big proportion of the effort that programmers put tin towards the design and implementation of the libraries should ideally be handled by the platform to the greatest extent possible.

Supabase is described as "an open-source Firebase alternative" (Supabase, n.d.). Some of the key features it offers among other services is a Postgres database, authentication, instant APIs, edge Functions and Storage. Described as a standard recipe for a BaaS platform, Supabase is different from Firebase at its foundation because it uses a relational database while Firebase uses a non-relational database. This gives the developers the ability to enforce structure at the database level. One other advantage of relational databases is the declarative nature of designing them as this allows the developers to specify what they want the database to do and not worry about how it does it.

In designing a relational database however, there is the learning curve of having to understand the Database Management System (DBMS) and the Structured Query Language (SQL) associated with it. Although, Supabase attempts to abstract this level of detailed knowledge, a developer developing a substantive system, would inevitably need knowledge of the DBMS and associated SQL to design a good backend on Supabase. SQL is a very backend-oriented technology/language and the author opines that one of the responsibilities of a BaaS platform ought to be to handle these backend implementation details to the greates extent possible. Hence a developer should expend most effort thinking about the frontend with a BaaS platform providing support for implementation of the backend functionality.

Supabase, when described as an open-source alternative attracts organizations and developers keen to take full control and self-host it although setting up Supabase on a self-hosted environment requires thorough knowledge of Linux servers, databases, networking, etc.

Having covered the dynamics of the concept of BaaS, this paper presents research done by the author towards designing and implementing a robust BaaS platform that addresses the major concerns of the platforms highlighted above in a way that is secure and meets elasticity demands of cloud platforms.

In the increasingly dynamic landscape of computing technology, it is crucial for businesses to develop solutions in a fast and efficient manner as they strive to maintain a competitive advantage by bringing their ideas into the market swiftly and cost-effectively. The need is not only to expedite the journey from concept to product but also to address the resource-intensive challenges of traditional software development lifecycle. The DbAPI BaaS platform designed with this in mind, has three key objectives as will be detailed later in this paper.

The first objective of the platform is to support acceleration of the development life cycle, ensuring features can be released faster hence giving several advantages to organizations. According to The Silver Logic (Marketing Team, 2022) - a company that specializes in building software solutions for their clients, fast software development;

1. Enhances client satisfaction and trust in services.
2. Allows developers to take on new projects, staying competitive in the industry.
3. Enables companies consistently produce clean, bug-free code quickly, earning a solid reputation hence attracting new customers.
4. Gives software development companies an edge over slower rivals by embracing speed.

Secondly, reducing software complexity is a crucial objective of the DbAPI BaaS platform. This goal entails simplifying aspects to do with DevOps i.e. developing, deploying, provisioning and maintaining backend systems running in production by minimizing infrastructure complexity. Organizations can achieve several benefits with this. Firstly, reduced infrastructure complexity also lowers the likelihood of introducing bugs in production that are infrastructure specific and improves overall software quality. Moreover, simpler infrastructure facilitates quicker on boarding of new team members as it is easy to understand how the software is deployed. This leads to increased productivity by saving time used to figure out how the infrastructure should work. This also promotes the use of agile software development practices and in essence, the reduction of infrastructure complexity is a strategic move that enhances the long-term viability and success of software projects within organizations.

The last, but not least objective is to reduce software development costs. By implementing efficient coding practices, the project aims to enhance productivity, ultimately leading to significant financial benefits to businesses.

2. Method

2.1 Approach

This section outlines the strategic approach employed to achieve the three objectives of the research project mentioned above. Focused on speeding up the transition from idea to a market-ready product, the methodology used to come up with DbAPI enables developers to save time. Additionally, it abstracts computing infrastructure, seeking to create simplify complexities of the back-end and reduce associated costs. This section details the systematic pathway adopted to realize the project's three key goals of efficiency, abstraction, and cost-effectiveness.

In order to speed up the development of software, the platform was designed to enable developers declare their system requirements without explicitly programming them. In declarative code, a programmer declares what needs to be done rather than describing how it should be done. The platform takes these requirements and automatically figures out a way to achieve them and by doing this it eliminates the need to write tedious control flow logic. This has the advantage of allowing the developers to focus on their ultimate software/product requirements, leaving the implementation details to the platform and effectively saving a lot of time.

Furthermore, the platform was designed to be usable without any special libraries. The BaaS platform relies on standard hypertext transfer protocol (HTTP) requests which can be made from any widely used programming language. Most programming languages natively support making of HTTP requests via its commonly used verbs such as *GET, POST, PUT*, *PATCH* and *DELETE.* These correspond to the various data operations supported by BaaS platforms namely *FETCH, CREATE, UPDATE, PARTIAL UPDATE* and *DELETE*. Having no libraries cuts down the time developers need to learn a new library that only works with a specific BaaS platform. This reduction in learning time can quickly become significant when one considers the time required to research and fix library-specific bugs. This is notwithstanding the fact that not all library specific bugs have solutions published and this means that on many occasions developers need to figure out solutions on their own. Taking all this into consideration, a BaaS platform which does not depend on any libraries can clearly save a lot of development time.

Moving to the second objective of reducing complexity, the DbAPI platform automatically manages computing infrastructure effectively reducing complexity of dealing with servers, networking, routing, uptime etc. The platform manages changing traffic by automatically implementing auto scaling of computing infrastructure. This ensures that the system does not slow down during high traffic allowing for elasticity to support optimum performance at all times.

This leads us to the third objective of the platform which is to reduce development and production costs of backend systems. Implementing declarative code and avoiding the requirement of using libraries implies that developers can move faster and get much more work done in a short time span. This also means that software development projects can be completed faster saving a lot of money which otherwise would be spent on hiring more developers to complete the project in a similar time span.

Reduction in usage of specialized infrastructure and expertise to set it up and maintain it further saves costs as organizations no longer need to invest in the infrastructure and talent to manage it. By supporting the DevSecOps approach, cost is also saved because the development team and the company does not need to worry about service uptime. This allows the company to work without having support staff available 24/7 ready to manage downtime, infrastructure failures and disasters.

2.2 Technology Description

By developing a robust platform that enables the author to attain the objectives mentioned above, the technology selected was crucial and played a pivotal role in shaping the platform. Architecting a solid framework that supports the building of such a robust platform lays a strong foundation over which high level abstractions can be built. At the core, there are three important design concepts; Concurrency of request handling, managed databases and back-end access control and security.

2.2.1 Concurrency and Parallel Processing

Concurrency accommodates multiple tasks that run and complete in overlapping time periods without a specific order (Bobrov, 2023). Parallel processing (parallelism) handles multiple tasks that run at the same time on a hardware with multiple computing resources like multi-core processor (Bobrov, 2023). **Figure 1** below shows the difference between these two technological concepts.

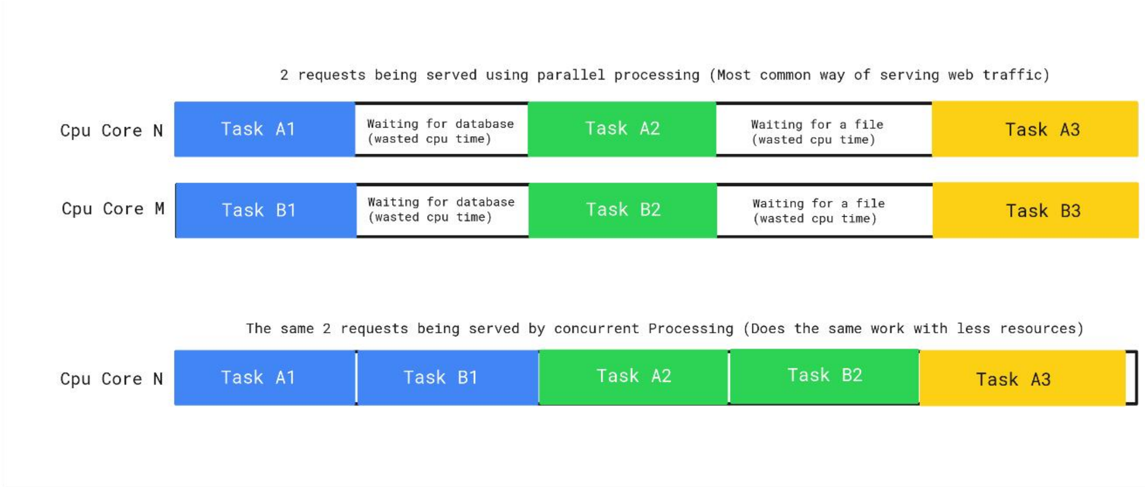

2 requests being served using parallel processing (Most common way of serving web traffic)
Cpu Core N
Task A1
Waiting for database (wasted cpu time)
Task A2
Waiting for a file (wasted cpu time)
Task A3
Cpu Core M
Task B1
Waiting for database (wasted cpu time)
Task B2
Waiting for a file (wasted cpu time)
Task B3
The same 2 requests being served by concurrent Processing (Does the same work with less resources)
Cpu Core N
Task A1
Task B1
Task A2
Task B2
Task A3

Figure 1. Concurrency and Parallelism

In **Figure 1**, the first section demonstrates how two requests of the same nature are served by a server employing parallel processing. Each request must complete tasks 1, 2 and 3 (where the output of task 1 is input to task 2 and the output of task 2 is an input to task 3) before sending a response to the user. The two requests require two different processes. In this case, the two processes run on two different central processing unit (CPU) cores. The CPU has to wait for the database server to respond between task 1 and 2 and also wait for the disk to seek and respond with a file between task 2 and 3. During this period, a system employing parallel processing will waste a lot of CPU time, requiring more resources to serve traffic.

Conversely, the lower portion of the figure demonstrates a configuration that serves the same two requests using concurrent processing. In this example, there is only one process running on a single CPU core. The process automatically picks up the second request (Task B1) while it waits for the database to respond to the first request. It then completes processing task B1 and moves to Task A2 since it has received the response from the database while it waits for the database response for the second request. This process continues for the rest of the tasks.

It is evident that the concurrent architecture illustrated above optimizes system resources unlike a parallel architecture.

Table 1 below summarizes the differences between concurrency and parallelism.

Table 1. Concurrency Verses Parallelism

Concurrency	Parallel Processing
Involves having a task running and managing multiple computations simultaneously.	Involves running multiple computations at the same time without having to manage them.
Can be done with a single processing unit.	Needs multiple processing units.
Increases the amount of work finished at a time.	Improves throughput and computational speed of the system.
Uses a non-deterministic flow approach.	Uses a deterministic flow approach.

For DbAPI, a concurrent architecture was selected to serve multiple requests. "It lets you finish a larger amount of work at a time when compared to parallel processing" (MKS075, 2020). This is great for *HTTP* types of systems like a BaaS platform where a lot of time is spent querying databases and waiting for items to be delivered over networks that may be resource constrained. It results in improved performance by utilizing system cores that spread out tasks. It allows for efficient scalability as developers are able to distribute workloads across multiple processors or nodes which are fully optimized.

2.2.2 Database Management

In managing the databases for the projects created on the BaaS platform, database transactions were chosen for all operations that create, update, or delete records. "A database transaction is a logical unit with several low-level steps such that if one step of the transaction fails, the whole transaction fails" (Zanini, 2023). If a transaction fails, all the steps are reversed leading to the database returning to its initial state through a process called a rollback. All this ensures that database transactions are Atomic, Consistent, Isolated and Durable (ACID).

Advantages of a database transaction include:

1. It allows for flexibility of the BaaS platform. This ensures users of the platform can perform their actions without having to change the overall structure of their project to make it stable.

2. Users can view history as the data is stored in a resource constrained environment.
3. There are minimal risks of losing data in the event of a datacenter power or system failure due to the database's consistency attribute.

Scalability in relational databases becomes complicated as to enable them scale vertically, they can vertically scale via upgrading Random Access Memory (RAM) and CPU to meet rising demands (LoginRadius, 2023). Hence scaling a relational database is expensive since acquisition of the components (RAM and CPU) required to support scaling raise cost. In order to mitigate this, the DbAPI BaaS platform was designed to implement auto scaling of databases so that during peak hours, resources are scaled up and conversely resources as scaled down during low traffic hours. This elasticity inherent in the platform enables users of the platform to benefit from the advantages of resources and avoid the costs that relate to the associated dynamic provisioning.

Data in the BaaS platform should also be stored in redundant database servers. This helps in making quick switches to the failover server in the event of a failure of hardware components such as drives. The databases of the BaaS platform should also have read replicas at different locations. Benefits of this include providing faster access speeds to data especially if a data center is closer to users physically. This can also help protect data by helping rebuild a database in the event of a failure to replace missing data from a nearby datacenter location.

2.2.3 Backend Access Control and Security

The backend part of a system is a really crucial part as it houses all the data that a user is working with. At all times access to the backend should only be given to people who have the right privileges and access controls.

BaaS platforms should have Role Based Access Control (RBAC) which is a measure put in place to restrict network access based on an individual's role within the organization to determine the level of access they get (Ellen, 2023). Features of the RBAC that the DbAPI BaaS system offers users include:

1. Management of role scope – allows limiting what areas the role group is allowed to access.
2. Management of role groups – Ability for a privileged user to add and remove members.

Benefits of implementing RBAC include improving security compliance, maximizing operational efficiency and reducing technical support and administrative work.

Another way of maintaining backend security for projects hosted on the BaaS platform is allowing access of the backend data through a token-based authentication. This is a system that allows users to verify their identity with a username and password on the hosted project and in return receive a unique access token that enables the user to access the website or webapp throughout the lifetime of the token without needing to provide their credentials each time (Okta, 2023).

There are 4 steps for obtaining a token to authenticate subsequent requests and these include:

1. Request access: User's client asks for access to a server.
2. Verification of credentials: Server authenticates user by checking their username and password.
3. Issuing of a token: The Server issues a token to the user's client.

4. The token is stored in the user's client and supplied to future requests to authenticate.

This is summarized in **Figure 2** below:

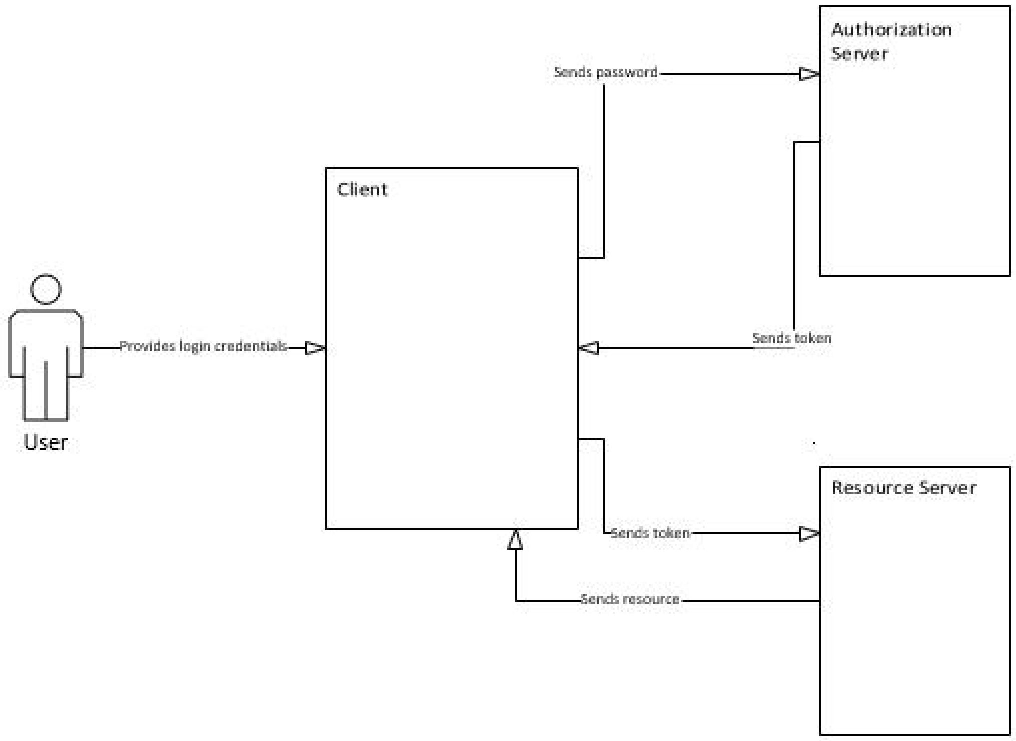


Figure 2. Token-based Authentication

Data encryption of sensitive items should be put in place for BaaS systems hence passwords and tokens should be hashed before any storage takes place. Hashing is a technique used to scramble raw information to the extent that it cannot reproduce its original form (Jena, 2023). Hashing algorithms like bcrypt and SHA-256/512 hash sensitive data that is stored in the backend of a system.

Bcrypt uses robust cryptography to hash and store passwords based on the Blowfish cipher (Chandramauliguptach, 2022). Bcrypt uses two functions to hash passwords, gensalt and hashpw. Gensalt generates a salt which is a pseudorandom string that is added at the end of the password before hashing thereby returning a pseudorandom string. Haspw creates a final hash that is stored in a database and can pass a salt and password in form of bytecode as arguments to it. This algorithm is also designed to be slow to avoid brute force attacks on password.

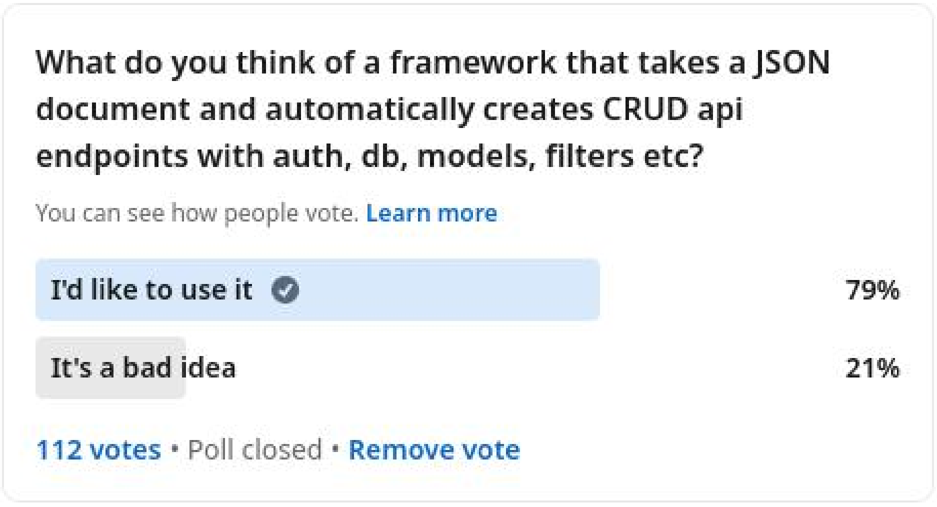
Secure Hashing Algorithm (Sha256) adds extra bits to the message such that its length is exactly 64 bits, a multiple of 512. This algorithm is designed to be much faster than bycrpt and this makes it suitable for hashing tokens. Since tokens need to be verified in every request, they need to be verifiable at high speeds. This does not provide opportunities for brute force attacks because tokens have a limited lifespan which is not long enough for a successful brute force attack. The length of the tokens is also much larger than passwords, further mitigating the risk of a successful brute force attack.

Other data that the user designates as sensitive should also be encrypted in the database. AES-256 should be used at the least to encrypt data. Encrypting data using this algorithm allows deciphering it when provided with the correct key. Advanced Encryption Standard (AES) is a block cipher with a key size ranging from 128/192/256 bits, that encrypts data in blocks of 128 bits each (randomsapien, 2023).

2.3 Developments

In order to evaluate the viability of creating a BaaS platform for developers, a comprehensive approach was undertaken to check the potential reception of the platform within the developer community. This was done in two steps. First, developer interest was gauged through a poll that attempted to figure out the inclinations of developers towards such a platform. Secondly, a tangible measure of developer enthusiasm was observed through an analysis of sign-ups on a waitlist subsequent to their interaction with the platform's waitlist website. This section goes over the results obtained by engaging with the developer community and the viability of the platform designed explicitly for developers.

Figure 3 below shows a screenshot for the poll conducted on LinkedIn's Django group/community to gauge the interest of developers. The Django community on LinkedIn was chosen because it was very active at the time of the poll and consisted of the vast majority of developers and software engineers/experts. Since the poll was done on a social media site, the question was posted in a way that uses easy terminology so that viewers of the poll are not overwhelmed by complex technical words on a social media site. The poll can also be viewed at LinkedIn Post.


What do you think of a framework that takes a JSON document and automatically creates CRUD api endpoints with auth, db, models, filters etc?

You can see how people vote. Learn more

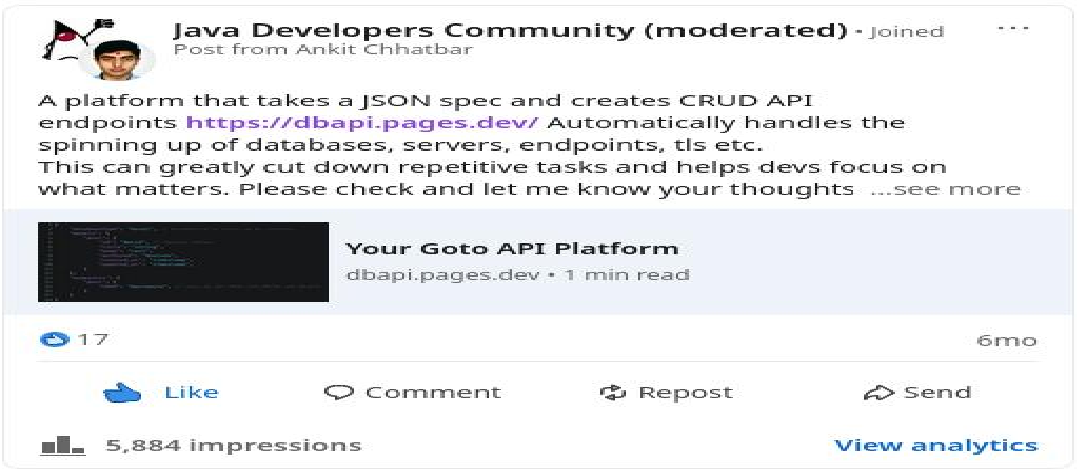
I'd like to use it ✓	79%
It's a bad idea	21%

112 votes • Poll closed • Remove vote

Figure 3. Results of Poll to Gauge Interest in the DbAPI Tool

The results of the poll were very favorable, consisting of 79% of the respondents keen to use such a platform.

Subsequently, members of the Python and Java groups on LinkedIn were also approached on what they thought of such a platform. The results were also very positive with almost 8,600 and 5,900 impressions from the respective groups. According to LinkedIn's help document, impressions "shows the number of times each post is visible for at least 300 milliseconds with at least 50 percent of the post in view on a signed in member's device screen or browser window" (LinkedIn, 2023). See **Figure 4** and **Figure 5** below for the responses received.



Java Developers Community (moderated) • Joined
Post from Ankit Chhatbar

A platform that takes a JSON spec and creates CRUD API endpoints https://dbapi.pages.dev/ Automatically handles the spinning up of databases, servers, endpoints, tls etc.
This can greatly cut down repetitive tasks and helps devs focus on what matters. Please check and let me know your thoughts ...see more

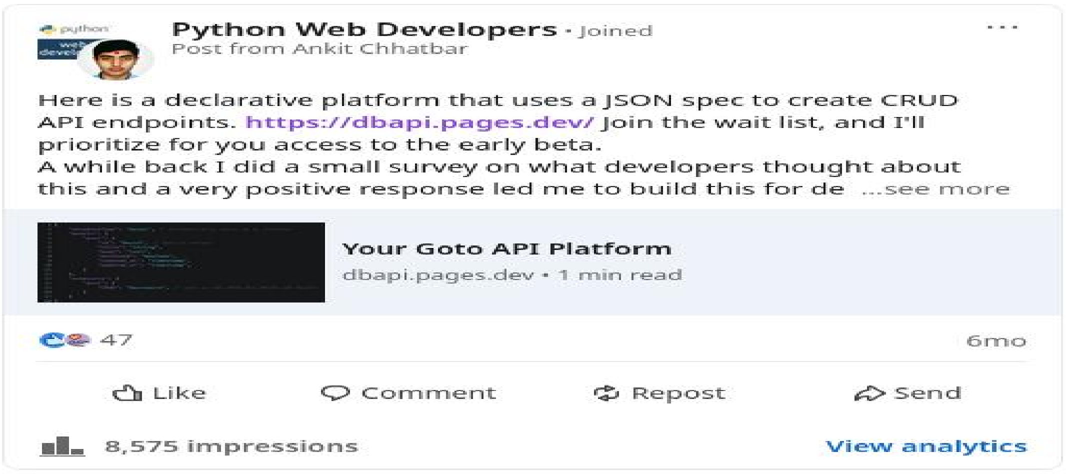
Your Goto API Platform
dbapi.pages.dev • 1 min read

17 6mo

Like Comment Repost Send

5,884 impressions View analytics

Figure 4. Responses from Java Developers Community



Python Web Developers • Joined
Post from Ankit Chhatbar

Here is a declarative platform that uses a JSON spec to create CRUD API endpoints. https://dbapi.pages.dev/ Join the wait list, and I'll prioritize for you access to the early beta.
A while back I did a small survey on what developers thought about this and a very positive response led me to build this for de ...see more

Your Goto API Platform
dbapi.pages.dev • 1 min read

47 6mo

Like Comment Repost Send

8,575 impressions View analytics

Figure 5. Responses from Python Web Developers Community

The above polls and posts also generated traffic to the waitlist website. **Figure 6** below is a graph showing the number of visitors to the site during this period. The author can confirm that these viewers were developers or software engineers because the spikes in traffic directly corresponded to the dates the communities were engaged on LinkedIn. The traffic estimates could potentially be higher than the figures shown below because many developers and software engineers use AdBlock which block capturing this data hence reducing the numbers.

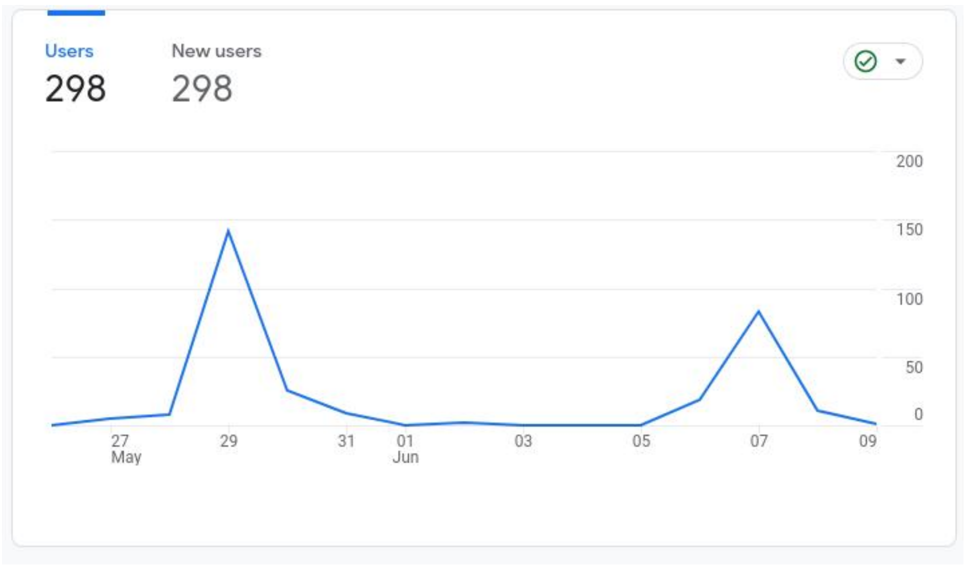


Figure 6. Number of Visitors to the Site from the Developer Community

Out of the almost 300 viewers shown above, 32 people (approximately 10%) signed up for the waitlist This could be considered reasonable because in order to sign up for the waitlist, users had to provide their personal details such as an email address and names to a website that they had only accessed for a minute or so.

3. Result

This section presents a comparison of the above named representative platforms with DbAPI in two aspects; i) Design of their respective APIs and ii) Utilization of the platforms.

The following sub sections provide a comparison for the process of designing the APIs on the three platforms.

Designing the API on:

3.1 Supabase

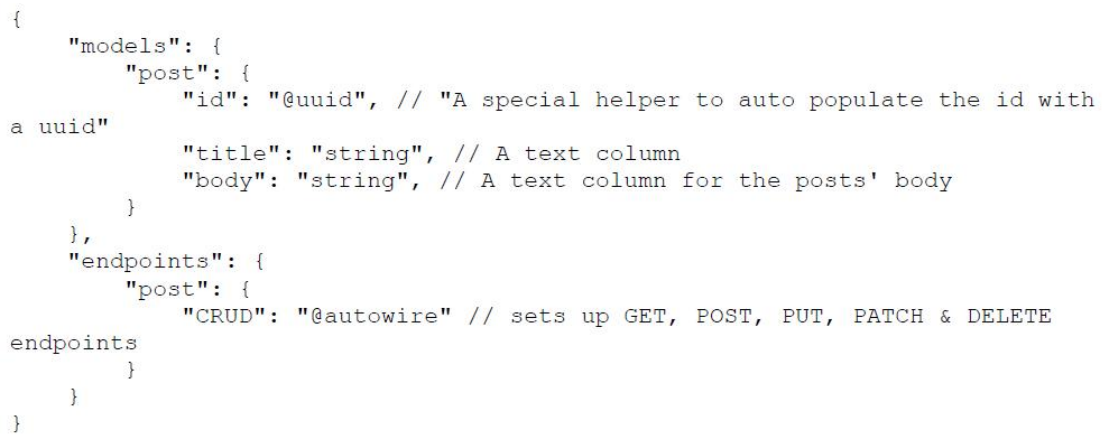
Involves in-depth knowledge of how relational databases work. Supabase enables developers to design a relational database using their user interface or SQL queries. Supabase then exposes the database as an API.

3.2 Firebase

Utilizes a product called Cloud Firestore described in Firebase's documentation as a "cloud-hosted, NoSQL database that your Apple, Android, and web apps can access directly via native Software Development Kit (SDK)". Setting Cloud Firestore as a backend requires creating a Firebase project and installing the relevant SDKs for each of the platforms a developer would like to develop on. One thing to note is that these are Firebase specific development kits and a developer needs to learn them in order to use them. They are not something available from the language.

3.3 DbAPI

Designing an API on DbAPI works by writing a specification automatically generates the database tables, endpoints and networking required to consume the API. Ultimately, DbAPI also automatically provisions servers and manages them in order to scale with the load received. **Figure 7** below illustrates an example specification to create an API endpoint to serve requests to blog posts on a blogging site.



```
{
    "models": {
        "post": {
            "id": "@uuid", // "A special helper to auto populate the id with
a uuid"
            "title": "string", // A text column
            "body": "string", // A text column for the posts' body
        }
    },
    "endpoints": {
        "post": {
            "CRUD": "@autowire" // sets up GET, POST, PUT, PATCH & DELETE
endpoints
        }
    }
}
```

Figure 7. Specification to Create an API Endpoint for a Blogging Site

The following sub sections provide a comparison for the process of using the respective BaaS platforms using the tools above.

Utilizing the Backend as a Service Infrastructure/Platform on:

3.4 Firebase

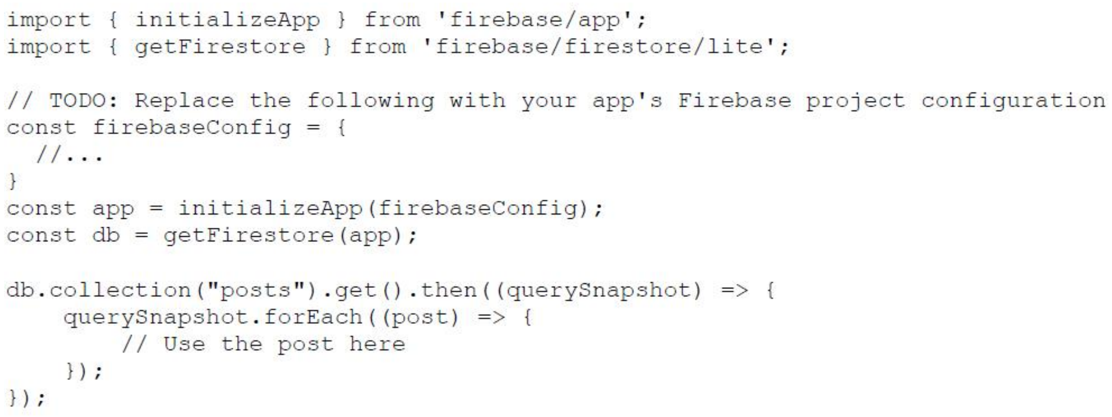
As mentioned earlier, Firebase requires platform specific libraries for each of the languages which the developer is required to learn in order to use Firebase. Below are examples in JavaScript, Python and Dart. Additionally, it should also be noted that the Firebase SDKs need to be installed before using it. **Figure 8** below shows code snippets required to retrieve blog posts from Firebase.

JavaScript

In order to consume a Firebase backend, first, a developer needs to install the SDK using the below commands;

1. npm install firebase
2. npm install firebase@10.7.1 --save

Below is the code to retrieve the posts;



```
import { initializeApp } from 'firebase/app';
import { getFirestore } from 'firebase/firestore/lite';

// TODO: Replace the following with your app's Firebase project configuration
const firebaseConfig = {
  //...
}
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

db.collection("posts").get().then((querySnapshot) => {
    querySnapshot.forEach((post) => {
        // Use the post here
    });
});
```

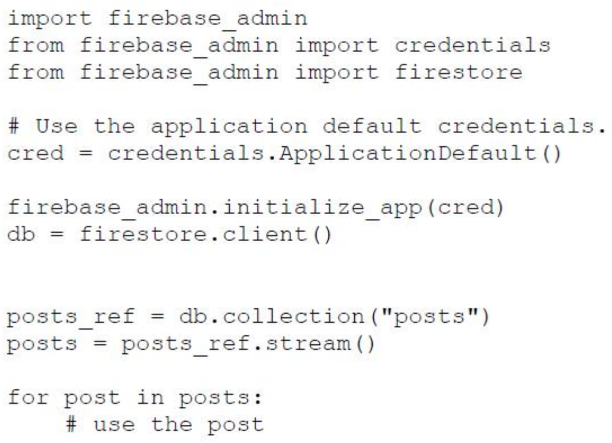
Figure 8. JavaScript Code Snippet to Retrieve Blog Posts

Python

In order to set up the Firebase SDK, a developer needs to:

1. Install the SDK using the command "pip install --upgrade firebase-admin"
2. Set up credentials to initialize Cloud Firestore (the steps to accomplish this are omitted to keep this paper short)

Figure 9 below shows code snippets to retrieve the posts from Firebase.



```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore

# Use the application default credentials.
cred = credentials.ApplicationDefault()

firebase_admin.initialize_app(cred)
db = firestore.client()


posts_ref = db.collection("posts")
posts = posts_ref.stream()

for post in posts:
    # use the post
```

Figure 9. Python Code to Retrieve Blog Posts

Dart/Flutter

In order to set up Firebase for flutter, the developer uses the below steps; It is important to note that the details in steps 1 and 2 have been omitted in this paper.

1. Install the Firebase Command Line Interface (CLI).
2. Install relevant dependencies.
3. Configure developed apps to use Firebase using the "flutterfire configure" command

Figure 10 below shows the code snippet to retrieve the posts after completing the above tedious setup process.

```
db = FirebaseFirestore.instance;

await db.collection("posts").get().then((event) {
  for (var post in event.docs) {
    // use the post
  }
});
```

Figure 10. Dart/Flutter Code Snippet to Retrieve Blog Posts

3.5 DbAPI

Unlike Firebase, the code used to utilize the API generated by DbAPI can be consumed without any special libraries that are specific to the BaaS platform. In other words, developers can use the standard approach of making HTTP requests in the language of their choice to consume the API. The below snippets show blog posts retrieved from the API using a *GET* request. Noteworthy is the fact that no DbAPI specific library is used. The domain used here is for example purposes. Hence the code shown in **Figure 11** below uses "example.com" which in reality is the domain that the developer sets.

JavaScript/NodeJS

Uses the fetch API which is natively supported in JavaScript;

```
fetch("https://example.com/api/posts")
    .then(response => response.json())
    .then((posts) => {
        // The posts can be used here
    })
```

Figure 11. JavaScript/NodeJS Code Snippet to Retrieve Blog Posts

Python

Uses the "requests" HTTP client which is developed by the Python Software Foundation - responsible for the development of the core Python distribution.

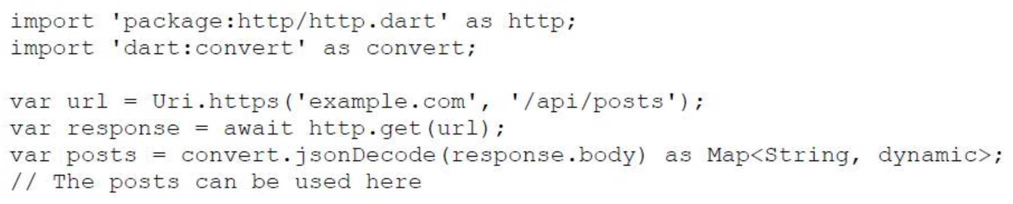
```
import requests

response = requests.get("https://example.com/api/posts")
posts = response.json()
# The posts can be used here
```

Figure 12. Python Code Snippet to Retrieve Blog Posts

Dart/Flutter

Uses the HTTP package which is the standard way of making HTTP requests in Dart/Flutter as shown in **Figure 13** below.



```
import 'package:http/http.dart' as http;
import 'dart:convert' as convert;

var url = Uri.https('example.com', '/api/posts');
var response = await http.get(url);
var posts = convert.jsonDecode(response.body) as Map<String, dynamic>;
// The posts can be used here
```

Figure 13. Dart/Flutter Code Snippet to Retrieve Blog Posts

4. Discussion

From the results of the above experiments, it is evident that it is simpler to use features available in languages natively instead of having a SDKs to achieve the same goals. The below are also added advantages that the author found when using the DbAPI platform instead of the others.

1. Faster development because of less code to get the same thing done.
2. Fewer bugs due to less code and less moving items in DbAPI compared to the others.
3. A less steep learning curve since developers already know how to make HTTP requests in the languages they use.
4. The time that the developer uses to learn the SDK can be used in actual development.
5. SDKs add an overhead to the application developed leading to lengthier startup and load times. The DbAPI platform has no SDK hence is faster in both cases.
6. Less data transferred for applications accessed over the network like web apps and websites because of the elimination of the need to transfer the SDK over the network in order for the application to run.

This project aimed to achieve several strategic objectives that promised huge business benefits across three main areas.

Firstly, by prioritizing the acceleration of the development life cycle, the platform seeks to enable organizations to release software features at a faster pace. This directly impacts client satisfaction and trust in services and also gives organizations a competitive edge by allowing developers to undertake new projects swiftly. Moreover, consistent delivery of clean, bug-free code not only fosters a solid reputation but also attracts new customers, establishing a significant advantage over slower competitors within the software development industry.

Secondly, a pivotal objective revolved around reducing software complexity, particularly in deploying and maintaining backend systems. Simplifying infrastructure complexities offers a myriad of benefits. It notably decreases the likelihood of introducing production-specific bugs, thereby elevating overall software quality. Additionally, streamlined infrastructure significantly expedites the on boarding process for new team members, saving valuable time that would otherwise be spent deciphering intricate systems. As mentioned earlier, this simplicity also promotes agile software development practices, thereby bolstering the long-term viability and success of software projects within companies.

Lastly, the author's efforts aimed at reducing software development costs by implementing efficient coding practices. Enhancing productivity through these practices leads to considerable financial benefits for businesses. Reduced time-to-market, minimized error correction efforts, and optimized resource utilization collectively contribute towards substantial cost reductions in the software development lifecycle. Together, achieving these objectives not only gives organizations a competitive advantage but also strengthens the foundation of software projects, fostering a more efficient, reliable, and cost-effective development environment for businesses.

This paper has presented the author's thoughts on Backend as a Service (BaaS) platforms, highlighting the strengths and limitations of widely used solutions like Firebase and Supabase while introducing the DbAPI platform as a novel approach. Beginning with a comprehensive overview of cloud services and BaaS models, it progresses to delineate DbAPI's objectives, methodologies, and technological underpinnings. By exploring technology choices such as concurrency, database management, and security measures, the paper elucidates the robust foundation of DbAPI. Engaging with the developer community through polls and waitlist sign-ups validates the platform's innovative approach. Ultimately, the paper emphasizes DbAPI's potential to accelerate development cycles, simplify complexities, and curtail costs, thereby heralding a promising era of agile and cost-effective backend development.

References

CloudFlare. (2023). *What is BaaS? | Backend-as-a-Service vs. serverless*. Retrieved from https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/

Douglass Ph.D, B. P. (2014). *Concurrency Architecture*. Retrieved from https://www.sciencedirect.com/topics/computer-science/concurrency-architecture#:~:text=The%20concurrency%20architecture%20identifies%20the,will%20be%20shared%20among%20them

Firebase. (n.d.). *Firebase*. Retrieved from https://firebase.google.com/

LoginRadius, T. (2023). *RDBMS vs NoSQL*. Retrieved from https://www.loginradius.com/blog/engineering/relational-database-management-system-rdbms-vs-nosql/#:~:text=Relational%20database%20or%20RDBMS%20databases,you%20scale%20by%20adding%20more

Marketing Team. (2022). *The Benefits and Disadvantages of "Fast" and "Slow" Software Development*. Retrieved from https://blog.tsl.io/the-benefits-and-disadvantages-of-fast-and-slow-software-development

Supabase. (n.d.). *Supabase*. Retrieved from https://supabase.com/

MKS075. (2020, November 25). *Difference between Concurrency and Parallelism*. Retrieved from https://www.geeksforgeeks.org/difference-between-concurrency-and-parallelism/

Gillis, A. S. (2021, December 21). *What is data redundancy?* Retrieved from https://www.techtarget.com/searchstorage/definition/redundant

Chandramauliguptach. (2022, June 03). *Hashing Passwords in Python with BCrypt*. Retrieved from https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/

Okta. (2023, February 14). *What Is Token-Based Authentication?* Retrieved from https://www.okta.com/identity-101/what-is-token-based-authentication/

Zanini, A. (2023, February 14). *Database Transactions 101: The Essential Guide*. Retrieved from https://www.dbvis.com/thetable/database-transactions-101-the-essential-guide/

Ellen, Z. (2023, May 5). *What is Role-Based Access Control (RBAC)? Examples, Benefits, and More*. Retrieved from https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more

randomsapien. (2023, May 22). *Advanced Encryption Standard (AES)*. Retrieved from https://www.geeksforgeeks.org/advanced-encryption-standard-aes/

Jena, B. K. (2023, August 29). *A Definitive Guide to Learn The SHA-256 (Secure Hash Algorithms)*. Retrieved from https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

LinkedIn. (2023, September). *Content analytics for your LinkedIn Page*. Retrieved from https://www.linkedin.com/help/lms/answer/a564051/content-analytics-for-your-linkedin-

Bobrov, K. (2023, November). *Concurrency vs Parallelism*. Retrieved from https://freecontent.manning.com/concurrency-vs-parallelism/