Event Ticketing Platform with Couchbase

FRANCISCO PRADA, MILENA GOUVEIA, SOFIA COSTA,

Faculdade de Engenharia da Universidade do Porto, Portugal

This report presents a concise investigation into the application of Couchbase technology within an event ticketing web platform. Focused on addressing event management challenges, the project explores Couchbase's suitability, emphasizing its flexibility, performance, and data handling capabilities. The report provides an overview of Couchbase's history, features, licensing, and comparative analysis with alternative solutions. Methodologically, it outlines the practical implementation of Couchbase in a prototype event ticketing platform. Key components include the conceptual data model, physical architecture, and data processing features. The prototype demonstrates Couchbase SDK integration, highlighting performance and flexibility benefits, alongside encountered limitations in analytics, transactions, and scalability. In conclusion, this project contributes to understanding Couchbase's potential in modern application development, particularly within different contexts, as well as other cases which benefit from a document-oriented architecture. Implications extend to developers and researchers exploring innovative database solutions.

CCS Concepts: • **Information systems** → **Database design and models**.

Additional Key Words and Phrases: NoSQL database, Couchbase, document database, event ticketing platform

ACM Reference Format:
Francisco Prada, Milena Gouveia, Sofia Costa. 2024. Event Ticketing Platform with Couchbase. In . ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In an era where digital transformation is ever-present, the selection of the right database plays a central role in the success of today's applications. Recently, NoSQL databases have shown to be the leading paradigm of this transformation due to their powerful flexibility when dealing with large amounts of data, high performance, and easier database handling for multiple and diverse use cases.

While exploring various NoSQL databases, our group opted for a document-based database solution, ultimately selecting Couchbase as our preferred database platform due to its popularity following MongoDB. In this report, we describe the exploration and integration of Couchbase for the development of an event ticketing web platform that addresses the needs not only of event organizers but attendees as well.

After an extensive description of the selected technology, from its history and main features to a comparative analysis with alternative solutions, we delve into the practical implementation of this database within a real-case scenario. We begin by providing an overview of the application's topic and dataset, followed by the presentation of its conceptual and physical data model. Next, we cover the implementation of Couchbase's data structures and processing features in our solution. Finally, we describe the architectural design of our prototype, demonstrating how Couchbase's features are integrated to meet the specific requirements of our application.

Manuscript submitted to ACM

2 COUCHBASE: TECHNOLOGY OVERVIEW

2.1 History

Couchbase, a leading NoSQL database, emerged from the merger of NorthScale and CouchOne in February 2011. NorthScale, founded in 2009, received early funding and later in 2010 rebranded as Membase, Incorporated. Meanwhile, CouchOne Inc., established in 2009 as Relaxed, developed and provided commercial support for Apache CouchDB open-source project, a document database [29].

Post-merger, Couchbase focused on building a scalable, high-performance document-oriented database system, marketed with the term NoSQL [29]. Significant investments, including a $14 million round led by Ignition Partners, fueled its growth. Couchbase garnered recognition, such as the 2012 Infoworld Bossie award, affirming its impact in the database landscape.

Transitions in leadership, including Peter Finter becoming chief marketing officer in 2016 and Matt Cain assuming the CEO role in 2017, underscored Couchbase's commitment to innovation.

2.2 Licensing

Couchbase offers different licensing options depending on the edition of their software. For developers and small projects, Couchbase provides the Community Edition under the Apache License, Version 2.0, enabling free use, modification, and distribution of the software. This edition serves as a simple platform for testing and development without the burden of licensing fees.

In contrast, Couchbase's Enterprise Edition targets larger use cases and enterprise environments, offering additional features, support, and maintenance updates. Enterprises benefit from a subscription-based licensing model, paying for each node or instance deployed, ensuring access to premium features and dedicated support.

Additionally, Couchbase offers Cloud Editions optimized for cloud-based deployments, enabling users to easily leverage Couchbase's capabilities within their preferred cloud infrastructure.

2.3 Pricing

Couchbase offers a variety of pricing options to suit different needs and use cases. For example, their Capella product offers an easy-to-start, fully managed solution with prices starting at $0.28 per node per hour for basic capabilities such as SQL++, search, indexing and more. Additional Capella plans, such as Developer Pro and Enterprise, offer enhanced features such as cross datacenter replication, analytics and eventing Prices start at $0.35 and $0.56 per node [1].

On the other hand, Couchbase Server offers enterprise clusters with support for SQL++, full-text search, and advanced security features, while Couchbase Mobile offers embedded, offline-first NoSQL databases with native sync capabilities. The pricing for these products is available upon request, allowing users to customize their plans based on their specific requirements.

Whether it's server deployments, mobile applications, or managed cloud services, Couchbase pricing options meet a wide range of applications, ensuring flexibility and scalability for all sizes of projects.

2.4 Community

With dedicated support from the Community Team and hundreds of developers, Couchbase offers a range of platforms and channels for developers to connect, learn, and share experiences [4]. The Couchbase Hub [2] serves as a central hub for receiving product updates, sharing projects, and staying informed about virtual and local events. Additionally,

developers can engage in real-time discussions and exchange ideas through the Let's Talk Couchbase Discord server, fostering a sense of community and camaraderie. For those seeking technical assistance or insights, the Couchbase forums [6] provide a valuable resource where engineers and experts offer timely responses and solutions.

2.5 Documentation

The Couchbase documentation [5] covers various aspects, especially on how to get started with Couchbase's products, whether you are new to this document database, updating, or migrating to it. It provides detailed instructions, tutorials, code samples, and references to help users understand and utilize Couchbase effectively.

Whether exploring Couchbase Capella for a fully managed database as a service, Couchbase Server for distributed document database needs, or Couchbase Mobile for edge computing applications, the documentation offers insights into deployment, management, scaling, and security aspects. Additionally, the documentation includes developer tools such as SDKs, connectors, command-line interface (CLI), and REST APIs, empowering developers to interact with Couchbase and integrate it seamlessly into their applications.

With a focus on user experience and accessibility, Couchbase's documentation enables users to quickly get started, troubleshoot issues, and explore advanced features, contributing to a smoother and more productive development experience.

3 MAIN FEATURES

Couchbase offers a range of powerful features that enable efficient data management, querying, synchronization, analytics, and event-driven processing [24]. Each feature is tailored to enhance productivity, optimize performance, and support real-time data operations. Here are the main capabilities that Couchbase supports:

- **SQL++**: Access NoSQL JSON data using a full implementation of SQL syntax, including support for joins and ACID transactions. SQL++ provides a seamless experience for developers familiar with SQL [19].
- **Key-value**: Efficient key-value access enables data retrieval via direct key lookup, optimizing performance for simple data access operations [9].
- **Full-text search**: Add a real-time search engine-like index to your data without data duplication or separate search tools, simplifying content search operations [13].
- **Analytics**: Perform complex business intelligence (BI) and analytics queries directly on Couchbase data with a powerful SQL implementation tailored for analytics purposes [16].
- **Eventing**: Execute custom code in response to data modifications using Couchbase's Eventing Service, facilitating near-real-time data mutation management and automation of business logic [12].
- **Transactions**: Perform a set of operations as a single unit of work, so that either all operations commit or roll back, which ensures data atomicity, consistency, isolation and durability (ACID properties). [23].
- **Cross Data Center Replication (XDCR)**: Synchronize data in real-time across different data centers with XDCR, ensuring data availability and supporting global distribution strategies with minimal latency [8].

4 DATA MODEL AND OPERATIONS

Couchbase provides a document data model based on JSON, whose internal structure consists of attribute-value pairs. This document model approach brings valuable advantages, such as flexibility, as it allows storage of documents with dynamic schema. [3].

Aside from that, the JSON documents used by this technology offer support for complex types, such as embedded documents and arrays. It enables each document to be self-contained, hence reducing the need to perform complex queries and join operations between them, which is a significant contribution to the rapid execution of application requests. [3]

This technology offers support for a set of data operations, which include:

- **Basic CRUD operations**: This refers to inserting, retrieving, updating, and deleting data from a cluster. [25]
- **Query Service using SQL++**: SQL++ (formerly known as N1QL) is a query language based on SQL that enables extraction and manipulation of JSON data. [21]
- **Indexing**: In order to reduce the response time to query and search operations, indexes can be created, representing a portion of data accessible for the search. Primary, Secondary, Full-Text and Analytics are the four forms of indexes offered by Couchbase. [14]
- **Searching**: Couchbase's Search Service offers Full-Text Search supported by appropriate indexes, providing diverse features, which include:
  - *"Language-Aware Searching"*: Capability to provide a specific word on the input and get results with variations of the word within the same language.
  - *"Geospatial Queries"*: Retrieve documents with geographic location.
  - *"Scoring Results"*: Return documents with the highest score.
- **Eventing**: Near-real-time data mutation management technique that involves defining operations to be executed when specific data modifications (events) occur, such as inserts, updates, deletes, and expiration. Some applications of the Eventing Service include propagating data changes inside a database, enriching documents instantly, and applying cascade delete to prevent orphaned records. [11, 15]

5 APPLICATION PROGRAMMING INTERFACES AND CLIENT LIBRARIES

Couchbase Server can be managed through Couchbase Web Console, the Command-Line Interface (CLI), and the REST API [17]. With these tools, users are able to add and configure nodes, define buckets for data storage, adjust settings, schedule backups, implement security measures, and perform Cross Data Center Replication (XDCR) for data availability. The Web Console offers a user-friendly interface for comprehensive cluster oversight, while the CLI and REST API provide scripting capabilities and command-line access for automation.

In addition, Couchbase provides a variety of Application Programming Interfaces (APIs) and client libraries to facilitate application development and integration with the Couchbase Server. These APIs are available in different programming languages SDKs, such as Java, .NET, Node.js, Python, Go, among other SDKs [22]. These SDKs provide developers with intuitive interfaces for performing various database operations, such as CRUD (create, read, update, delete), querying, and data manipulation.

6 CONSISTENCY AND REPLICATION FEATURES

As mentioned previously, Couchbase provides features like Cross Data Center Replication (XDCR), which enables the replication of data across clusters located in different data-centers [10]. Inter-cluster replication is used to ensure equal distribution and high availability of data. The protocol used for XDCR is called DCP (Data Change Protocol). This protocol is also used for inter-cluster replication and, in both cases, is used for low-latency memory-to-memory replication. Furthermore, Couchbase offers flexible conflict resolution mechanisms, such as sequence-based and timestamp-based

resolution, enabling organizations to manage conflicts efficiently and maintain data coherence even in scenarios of concurrent updates.

7 DATA PROCESSING FEATURES

As detailed in Section 3, Couchbase offers a comprehensive set of data processing features designed to address a wide range of use cases and scenarios. These features include the N1QL query language which is familiar to the SQL syntax. Additionally, Couchbase offers a full-text search engine that allows users to perform text-based searches across their data efficiently. The NoSQL database Analytics functionality enables direct querying of analytical data without impacting performance, although this feature is behind a paywall and consequent upgrade to the Enterprise Edition. Furthermore, the Eventing Service which allows users to define custom business logic that executes in real time in response to data changes. Couchbase also provides tools for data transformation and manipulation, allowing organizations to preprocess data before storing it in the database or before serving it to applications. Finally, Couchbase integrates with various data processing and analytics tools, including Apache Kafka, Apache Spark, and Elasticsearch. Couchbase also used to support in older versions the generation of Views, which were used to create MapReduce functions.

8 USE CASES AND SCENARIOS

Couchbase empowers organizations of every kind to thrive in today's various industries by providing exceptional user experiences and low cloud operational costs [20]. By choosing Couchbase as their noSQL solution, businesses can address a wide range of use cases tailored to their specific needs. These use cases are described as the following:

- **Fix Slow Application Performance**: Couchbase helps organizations enhance application responsiveness by implementing caching mechanisms, ensuring low latency response times, and maintaining high availability, thereby improving overall user satisfaction.
- **Enable Distributed Workloads**: With Couchbase's clustering and replication features, businesses can achieve global availability and fault tolerance while meeting regional data compliance requirements, enabling seamless distributed workloads.
- **Improve Application Flexibility**: Couchbase empowers organizations to easily adapt to evolving user needs during both development and deployment phases. Its flexible JSON data model and robust multi-model data access services facilitate agile application development and deployment.
- **Create Mobile, Edge, and IoT Apps**: Couchbase enables the development of robust mobile, edge, and IoT applications by providing features like offline data synchronization, embedded database support, and automatic device syncing, ensuring seamless data access and performance optimization, even in disconnected environments.
- **Boost Developer Productivity**: Couchbase empowers developers by allowing them to choose their favorite programming languages, to build and deploy applications more efficiently, resulting in faster time-to-market and increased productivity.
- **Reduce the Cost of Operations**: By consolidating technology stacks, optimizing price-performance ratios, and streamlining development processes at scale, Couchbase helps businesses lower operational costs and complexity.

The use cases mentioned cater to the unique needs of various industries such as Energy and Utilities, Financial Services, Gaming, Healthcare, Retail and E-commerce. Media and Entertainment, Telecommunications, Travel and Hospitality, Manufacturing and Logistics and Government.

9 COMPARISON TO OTHER SOLUTIONS

In this section, we compared Couchbase to other document-oriented NoSQL databases such as MongoDB [28] and CouchDB [26].

9.1 Couchbase vs. MongoDB

MongoDB stands out as a NoSQL database trusted by large enterprises and used for a wide variety of use cases. In terms of features, MongoDB offers multi-document ACID transactions, continuous backup with point-in-time recovery, client-side field-level encryption, built-in full-text search, and a robust ecosystem. On the other hand, Couchbase offers limited support for ACID transactions and lacks continuous backup capabilities, and does not offer client-side field-level encryption [27].

Performance-wise both platforms are fast compared to competitors however, MongoDB is preferred by developers due to its simplicity, unified query language, and consistent secondary indexes while Couchbase is criticized for its complexity and limited adoption.

9.2 Couchbase vs. CouchDB

The official Couchbase website offers a brief comparison between Couchbase and CouchDB [7], mentioning the main challenges with CouchDB, the difference in features, and feedback from customers.

The three main challenges identified with CouchDB are the lack of data consistency, performance speed and support for development tools. In terms of features, Couchbase offers strong data consistency with distributed ACID transactions, master-master replication, advanced query language support with SQL++, and comprehensive services including data, query, index, full-text search, analytics, eventing, backup, and mobile sync. On the other hand, CouchDB provides eventual consistency, limited query capabilities, and lacks support for some essential features like distributed ACID transactions.

When it comes to performance at scale, Couchbase excels with features like integrated and managed cache, automatic failover, and cross-data center replication, providing a more robust and scalable solution compared to CouchDB.

10 PROTOTYPE

10.1 Topic and Dataset Overview

The prototype presented below serves as an example of the utilization of Couchbase's document-based technology to enhance the organization and efficiency of a web platform tailored for event ticketing. This endeavor represents a concerted effort to harness the capabilities of Couchbase, facilitating seamless data management and retrieval for both event organizers and attendees. This platform caters to diverse event categories, ranging from art exhibitions to music concerts. The dataset utilized for this purpose is structured into six distinct classes, each serving a specific role in the platform's ecosystem:

- **Artists**: This class encompasses 185 real-life music artists, each identified by a unique name and ID. Additionally, artists are supplemented with a randomly generated biography. Furthermore, details of their top five songs and a list of event IDs they are scheduled to attend/attended are included for comprehensive artist profiling.
- **Categories**: Comprising 80 distinct event categories, this class provides a taxonomy for organizing events on the platform. Each category is identified by a unique ID and name, with associated lists of events. This hierarchical structure facilitates intuitive navigation and content discovery for users.

- **Events**: At the core of the dataset lies an aggregate of 3,850 events, encompassing all essential information. Each event is characterized by a unique ID and name, along with details such as the organizing entity's ID, a description aligned with its category, and the event date in ISO format. Venue information, including the address and location, is provided for logistical convenience. User engagement metrics, such as like counts, offer insights into event popularity, while ticket details, including types and prices, facilitate seamless ticket purchasing.
- **Locations**: This class serves to catalog event venues, providing essential information for event planning and navigation. Each location entry includes an ID and name, along with a list of event IDs scheduled to take place at the respective venue. This association ensures efficient event-location mapping and enhances user experience.
- **Transactions**: Used for tracking user interactions and transactions. Transactions are categorized into two states: **purchased** and **shopping_cart**, signifying completed purchases and items in the shopping cart, respectively. Each transaction record includes a unique ID, timestamp, status flag, user ID, and an itemized list detailing event and ticket information.
- **Users**: User profiles are integral to the platform's functionality, serving as the gateway to personalized experiences. Each user entry features a unique ID, name, username, email, and password (for simplicity's sake during testing, the generated users all have 'password' as their password). A flag denotes organizational affiliation, while a list of liked events and their corresponding IDs caters to individual user preferences. This last one only shows if the user is not an organization.

The dataset was generated using tools such as **ChatGPT** and the **Faker** Python package. The accompanying scripts, located in the 'data' folder within the 'src' directory, facilitate the generation and organization of data across various classes, ensuring a cohesive and representative dataset for prototype development.

10.2 Conceptual Model and Implemented Data Structures

The entities and the relation between them in represented in Figure 1 in the Appendix A. Each event takes place in a certain city and is classified into multiple categories. Some events also have artists associated with them. Furthermore, events have multiple comments and at least one type of ticket that can be purchased.

The database stores two distinct types of users: consumers and organizations. Both have the ability to browse, search and filter events. However, only the consumers can purchase tickets for events, like events and evaluate them by sending comments. Organizations, on the other hand, have the capacity to host events.

To manage the relationships between the described entities, we organized the data into six collections: locations, categories, artists, events, users and transactions. Some collection, such as events, users and transactions have nested information in order to simplify information retrieval and ensure that related data is stored together. Each document in the events collection includes nested structures for location, categories and ticket types (see Listing 1 in the Appendix B). Furthermore, this collection exhibits a heterogeneous schema, since some events may contain a list attribute "artists" (see Listing 2 in the Appendix B). Similarly, in the users collection, comments are nested within each document so that users can easily read, edit and delete the comments they made directly from their profile. It also ensures that the user's information, such as their name, is automatically updated whenever their personal information is modified (see Listing 3 in the Appendix B). Finally, the transactions collection contains nested information related to individual items within each transaction, which allows us to store detailed information of the event name, price and quantity of each type of ticket (see Listing 4 in the Appendix B).

10.3 Architecture and Physical Data Model

The prototype was developed using React.js to build the frontend, managing data presentation and user interactions. On the other hand, the backend was built using Node.js, incorporating the Couchbase SDK to interact with the Couchbase database. It exposes multiple endpoints that enable communication with the frontend, facilitating data storage, retrieval and manipulation. The Physical Data Model, presented in Figure 2 in the Appendix C, illustrates the communication between the frontend and the backend using HTTP requests.

10.4 Data Processing Features

As previously discussed, Couchbase technology provides a wide array of options and features for data processing. The following section provides a detailed breakdown of each utilized feature, along with insights into potential limitations encountered during implementation or attempted implementation:

- **N1QL Querying** - Couchbase employs N1QL as its querying language, a syntax closely resembling SQL, which renders it familiar to all members of the development team. Throughout the application's development, the querying feature proved indispensable, serving a multitude of purposes. It was utilized extensively to retrieve information across various classes, such as facilitating user logins, accessing event details, retrieving user shopping carts, and more. Additionally, N1QL queries were instrumental in executing data manipulation tasks, including user registration, deletion, and managing shopping cart items. Moreover, the querying capability extended to filtering and grouping data, notably powering the filter options prominently featured on the homepage (see Figure X).
- **Indexing** - The indexing service provided by Couchbase played vital role in improving query efficiency, particularly in queries involving sort and group operations, such as the filters in the homepage, the event comments and the display of purchased tickets organized into "past" and "upcoming" sections. This was crucial to ensure quick response times and a smooth user experience.
- **Full-Text Search** - The Couchbase technology offers developers an easily accessible method for implementing a FTS feature via its WebUI. By activating the "Search" service within the cluster, developers can swiftly create a Search Index. This index allows for the selection of attributes to search, application of **tokenizers**, and implementation of various **filters** tailored to the search criteria, including language and morphology considerations. In our implementation, we utilize an **English Snowball stemmer**, which employs Natural Language Processing (NLP) techniques to strip suffixes from terms, enhancing search accuracy. Additionally, an **English plural stemmer** is employed to normalize plural words to their singular form, ensuring consistency in search results. We configure the search to query specific attributes such as event names, descriptions, and artist lists. Notably, the search on descriptions is set to the English language, while others are marked as "standard." After rigorous testing, we generate the search index and obtain a CURL command for seamless integration into the cluster setup via the terminal.

As previously mentioned, while there were other Couchbase features worth exploring, limitations were encountered during their implementation. One such feature was **Analysis**, intended to provide insightful data for organizations to gauge the success of their events and overall business management. However, due to the exclusivity of the Analysis feature to paid versions of Couchbase[16], manual querying was necessary to obtain these results. Another feature that was considered but ultimately discarded was **MapReduce Views**, which were also intended for analytics purposes, such as analyzing an organization's events. However, this feature is no longer supported, and the ability to create

them is limited to the default collection only. Despite efforts to find alternatives, the group was unable to identify a viable solution, especially concerning the NodeJS SDK[18]. Lastly, **ACID Transactions** was also considered utilizing to perform ticket purchases, so that the quantity of available tickets could be updated whenever a new purchase was made. However, our study revealed that this feature is only available in Couchbase Server version 7.2 and can only be accessed through the Couchbase Web Console, with no support for Node.js SDK in Javascript. To overcome this challenge, it was decided to have unlimited tickets for all events. It's also worth noting that Couchbase provides a multi-node solution and it is not single-node focused like other technologies. However, due to some complications with the Docker configuration of the project's cluster, every implemented feature was centered in only one node, when the initial and ideal solution was to distribute them between different nodes.

11 CONCLUSION

This report has provided a detailed exploration of Couchbase technology, showcasing its evolution, features, capabilities, and practical implementation in a real-world scenario. As demonstrated, Couchbase emerges as a robust NoSQL database solution, offering a range of powerful features such as SQL++, full-text search, analytics, eventing, and cross-data center replication. Its **document-based** data model, coupled with an extensive community support, makes it an attractive choice for organizations across various industries. Through the prototype implementation of an event ticketing web platform, we have highlighted how Couchbase's features can be leveraged to address specific use cases, streamline data management, and enhance user experiences. Despite encountering certain limitations during implementation, such as the availability of certain features only in paid versions or compatibility issues with specific SDKs, Couchbase's overall versatility and performance capabilities remain evident. Looking ahead, Couchbase continues to evolve, with ongoing developments and advancements aimed at enhancing its functionality, scalability, and security. With its proven track record and growing adoption, Couchbase remains a leading player in the NoSQL database landscape, empowering organizations to meet the demands of modern applications and digital transformation initiatives effectively.

ACKNOWLEDGMENTS

The group would like to express their sincere gratitude to the course's professor, Sérgio Sobral Nunes, for his continuous support and guidance throughout the project. His availability and expertise have been pivotal in the successful completion of this endeavor. We also extend our appreciation to the Couchbase community for their invaluable support through detailed documentation and insightful forums. Their contributions, albeit indirect, have greatly enriched our understanding and implementation of Couchbase technology.

REFERENCES

[1] Couchbase. 2023. *Couchbase Pricing for Capella, Server + Mobile Subscriptions.* https://www.couchbase.com/pricing/ [Accessed on March 29, 2024].
[2] Couchbase. 2024. *Couchbase Community Hub.* https://community.couchbase.com/ [Accessed on March 29, 2024].
[3] Couchbase. 2024. *The Couchbase Data Model.* https://docs.couchbase.com/server/current/learn/data/document-data-model.html [Accessed on April 24, 2024].
[4] Couchbase. 2024. *Couchbase Developer Community.* https://www.couchbase.com/developers/community/ [Accessed on March 29, 2024].
[5] Couchbase. 2024. *Couchbase Documentation | Couchbase Docs.* https://docs.couchbase.com/home/index.html [Accessed on March 29, 2024].
[6] Couchbase. 2024. *Couchbase Forums.* https://www.couchbase.com/forums/ [Accessed on March 29, 2024].
[7] Couchbase. 2024. *Couchbase vs. CouchDB: Differences Between Them.* https://www.couchbase.com/comparing-couchbase-vs-couchdb/ [Accessed on May 3, 2024].
[8] Couchbase. 2024. *Cross Data Center Replication (XDCR) | Couchbase.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].
[9] Couchbase. 2024. *Data | Couchbase Docs.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].

[10] Couchbase. 2024. *Data Center Replication and Synchronization | Couchbase.* https://www.couchbase.com/blog/couchbase-replication-and-sync/ [Accessed on May 1, 2024].
[11] Couchbase. 2024. *Eventing Service.* https://docs.couchbase.com/server/current/learn/services-and-indexes/services/eventing-service.html [Accessed on April 25, 2024].
[12] Couchbase. 2024. *Eventing Service: Features and Capabilities - Couchbase.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].
[13] Couchbase. 2024. *Full-Text Search (FTS) | Couchbase.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].
[14] Couchbase. 2024. *Indexes.* https://docs.couchbase.com/server/current/learn/services-and-indexes/indexes/indexes.html [Accessed on April 24, 2024].
[15] Couchbase. 2024. *Introduction to Eventing.* https://docs.couchbase.com/server/current/eventing/eventing-overview.html [Accessed on April 25, 2024].
[16] Couchbase. 2024. *JSON Analytics + Real-Time Insights | Capella Columnar.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].
[17] Couchbase. 2024. *Manage Overview | Couchbase Docs.* https://docs.couchbase.com/server/current/manage/management-overview.html [Accessed on May 1, 2024].
[18] Couchbase. 2024. *MapReduce Views.* https://docs.couchbase.com/nodejs-sdk/current/howtos/view-queries-with-sdk.html [Accessed on May 13, 2024].
[19] Couchbase. 2024. *N1QL | SQL-Like JSON Database Query Language | SQL++.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].
[20] Couchbase. 2024. *NoSQL Use Cases Across Industries | Couchbase.* https://www.couchbase.com/use-cases/ [Accessed on April 30, 2024].
[21] Couchbase. 2024. *Query Data with SQL++.* https://docs.couchbase.com/server/current/n1ql/query.html [Accessed on April 24, 2024].
[22] Couchbase. 2024. *SDK Extension Libraries | Couchbase Docs.* https://docs.couchbase.com/sdk-extensions/index.html [Accessed on May 1, 2024].
[23] Couchbase. 2024. *Transactions.* https://docs.couchbase.com/server/current/learn/data/transactions.html [Accessed on May 13, 2024].
[24] Couchbase. 2024. *What Is Couchbase | Couchbase Developer Portal.* https://developer.couchbase.com/what-is-couchbase/ [Accessed on March 29, 2024].
[25] Couchbase. 2024. *Work with Documents.* https://docs.couchbase.com/server/current/guides/kv-operations.html [Accessed on April 24, 2024].
[26] CouchDB. 2024. *Apache CouchDB.* https://couchdb.apache.org/ [Accessed on May 3, 2024].
[27] MongoDB. 2024. *Couchbase Vs MongoDB | Differences Use Cases | MongoDB.* https://www.mongodb.com/resources/compare/couchbase-vs-mongodb [Accessed on May 3, 2024].
[28] MongoDB. 2024. *MongoDB: The Developer Data Platform | MongoDB.* https://www.mongodb.com/ [Accessed on May 3, 2024].
[29] Wikipedia. 2023. *Couchbase, Inc.* https://en.wikipedia.org/wiki/Couchbase,_Inc. [Accessed on March 29, 2024].

A CONCEPTUAL MODEL

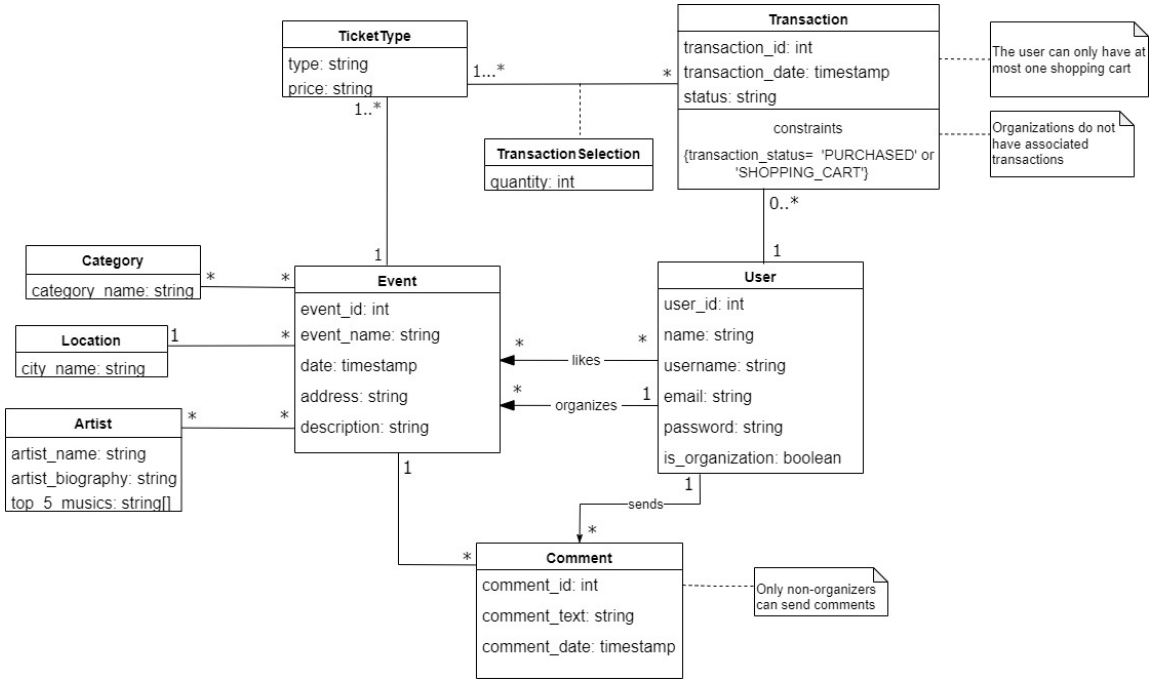

Fig. 1. Conceptual Model

B DATA STRUCTURES

```
{
    "event_id": "1",
    "organization_id": "1536",
    "event_name": "Magical Shopping Marathon",
    "description": "Challenge yourself and join us for an exhilarating marathon.",
    "date": "2024-02-02T16:00:00",
    "address": "088 Stephanie Corners Apt. 322\nCervanteschester, MH 23514",
    "location": "Indio",
    "categories": ["Retail", "Shopping", "Sales"],
    "num_likes": 1,
    "ticket_types": [
        {
            "ticket_type": "5K Race",
            "price": 40
        },
        {
            "ticket_type": "10K Race",
            "price": 50
```

```
        }
    ]
}
```

Listing 1. Example of a document in the 'events' collection

```
{
    "event_id": "81",
    "organization_id": "1553",
    "event_name": "Radiant Music Exhibition",
    "description": "Explore the beauty of creativity at our exhibition.",
    "date": "2024-08-24T21:30:00",
    "address": "6146 Bolton Green\nSmithhaven, TX 11347",
    "location": "Novosibirsk",
    "categories": ["Entertainment", "Concert", "Music"],
    "num_likes": 2,
    "ticket_types": [
        {
            "ticket_type": "Standard Entry",
            "price": 40
        },
        {
            "ticket_type": "VIP Experience",
            "price": 100
        }
    ],
    "artists": ["H.E.R.", "Wolf Alice", "Bon Iver"]
}
```

Listing 2. Example of a document with the "artists" attribute in the 'events' collection

```
{
    "user_id": "17",
    "name": "Dillon Stark",
    "username": "dillon_stark",
    "email": "dillon_stark@example.com",
    "password": "password",
    "is_organization": false,
    "liked_events": ["163", "2263", "1297"],
    "comments": [
        {
            "comment_id": "515",
            "date": "2023-12-22T22:31:14.788756",
            "text": "Comment text",
            "event_id": "1695",
```

```
            "event_name": "Dazzling Crafts Webinar"
        },
        {
            "comment_id": "3075",
            "date": "2023-12-09T08:20:33.655983",
            "text": "Comment text",
            "event_id": "267",
            "event_name": "Marvelous Crafts Festival"
        }
    ]
}
```

Listing 3. Example of a document in the 'users' collection

```
{
    "transaction_id": "1",
    "transaction_date": "2024-02-08T20:48:50",
    "transaction_status": "purchased",
    "user_id": "1369",
    "items": [
        {
            "event_id": "91",
            "event_name": "Whimsical Technology Seminar",
            "event_date": "2024-10-06T18:00:00",
            "ticket_type": "Regular Ticket",
            "ticket_price": 50,
            "quantity": 1
        },
        {
            "event_id": "2703",
            "event_name": "Innovative Literature Webinar",
            "event_date": "2024-03-15T04:30:00",
            "ticket_type": "Free Admission",
            "ticket_price": 0,
            "quantity": 2
        }
    ]
}
```

Listing 4. Example of a document in the 'transactions' collection

C PHYSICAL DATA MODEL

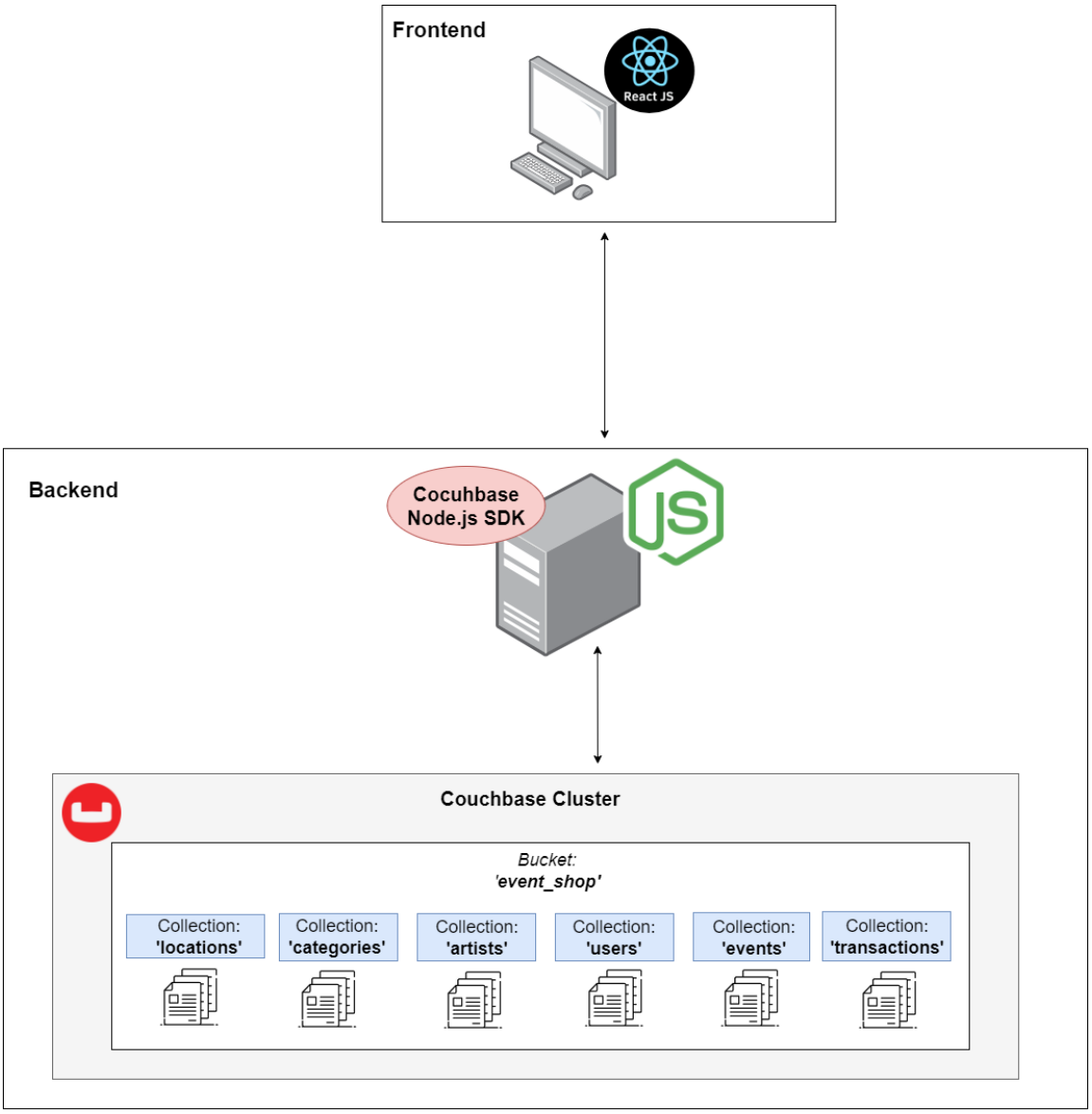

Fig. 2. Physical Data Model

D PROTOTYPE

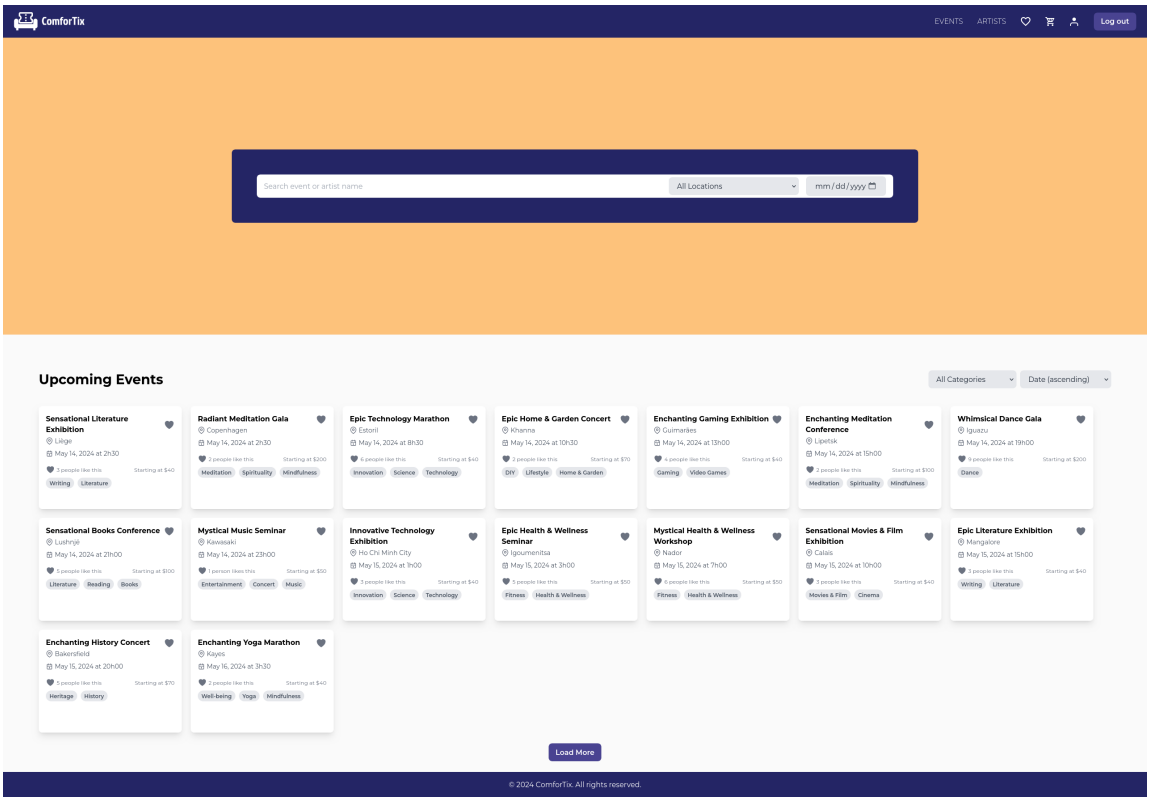

Fig. 3. ComforTix Homepage

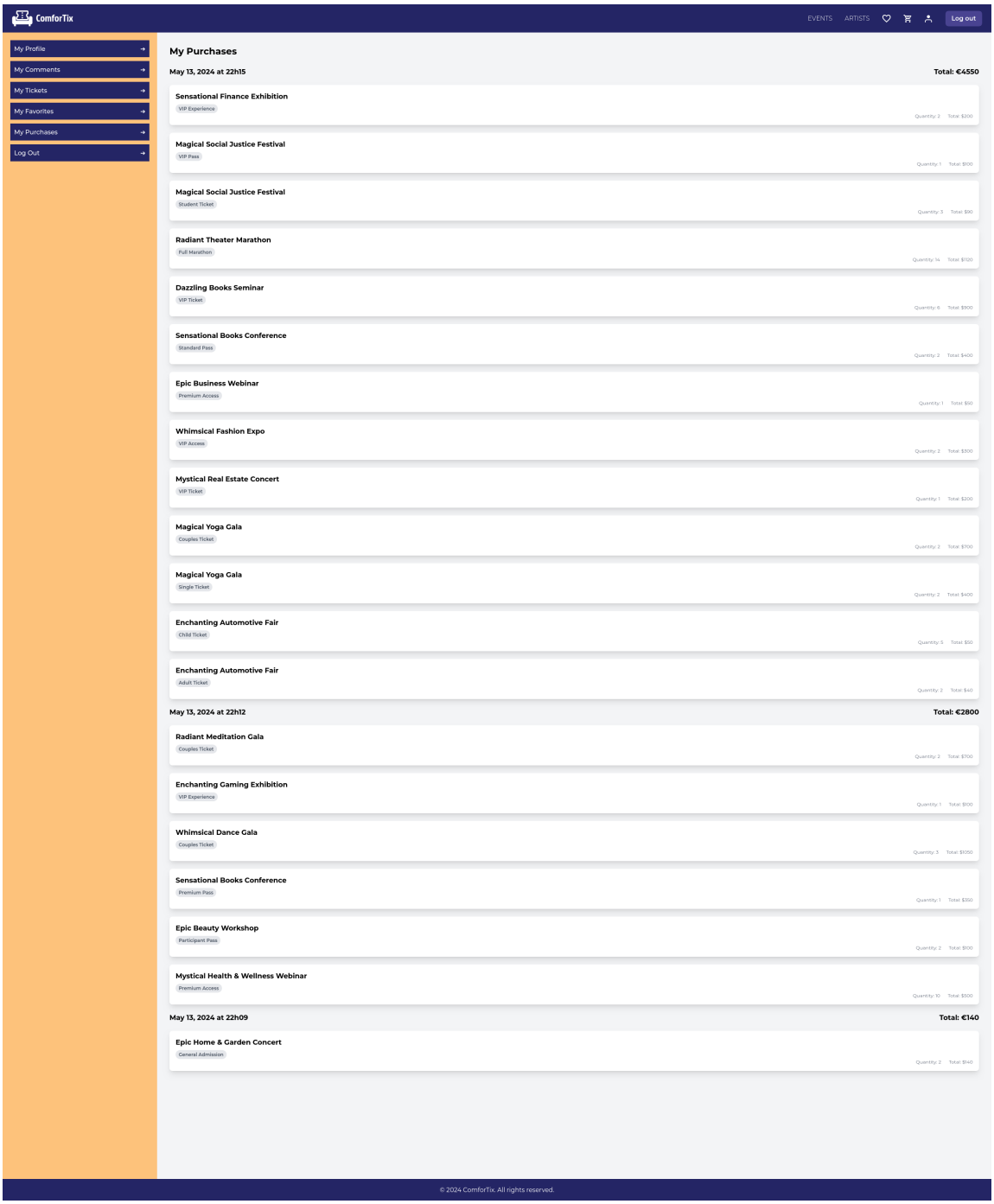

Fig. 4. Users purchases per day and money spent

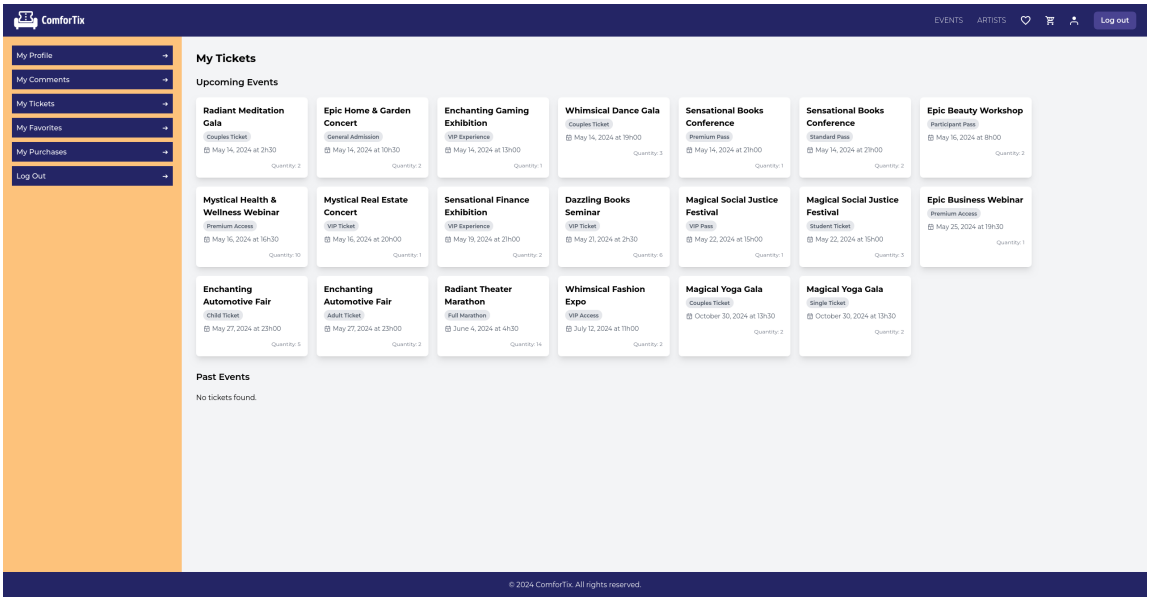

Fig. 5. Users purchased tickets