Tech Phantoms

Josemon Baby	Grishma Kagde	Yash Sowani
<22200681>	<22200378>	<22200139>

Synopsis:

Creating a system that makes it easier to transfer a URL, text, or document from one device to another. There are several other solutions such as Google Drive and other cloud apps, but the main focus here is to eliminate the requirement for creating an account, downloading their software, or maintaining sync for particular files or folders. A Spring Web Application using REST that could be launched as a website and accessed by both PCs and smartphones would eliminate needless time wastage. This web based application will not require registration, just upload the file and immediately after it is uploaded, a QR code is generated that can be scanned by a smartphone and a shortened URL is available that can be used in another PC to direct users to the resource's download landing page.

Technology Stack:

- **REST** : REST (Representational State Transfer) is an architectural style for building APIs (Application Programming Interfaces). REST is popular because it is simple and easy to use, and it is an efficient way to expose data and functionality through APIs. The foundation of REST is the concept of resource-oriented architecture, which places more emphasis on resources and the actions that can be taken on them than on particular endpoints or methods. Ideally suited for usage in cloud-based, microservice-oriented systems.

- **SpringBoot** : Spring Boot is a popular Java-based framework used for building web and enterprise applications. It is built on top of the Spring framework, and it is designed to make it easier to create stand-alone, production-grade Spring-based applications. Spring Boot is highly modular and provides support for monitoring and metrics, security, and externalised configuration.

- **Hibernate**: Hibernate ORM is an object–relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object–relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

- **MySQL**: MySQL Cluster is the distributed database combining linear scalability and high availability. It provides in-memory real-time access with transactional consistency across partitioned and distributed datasets. It is designed for mission critical applications.

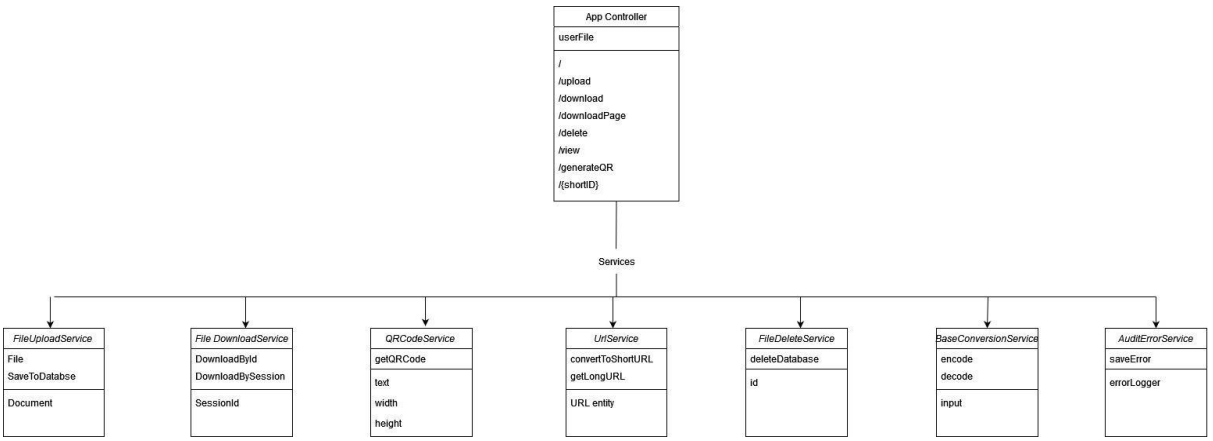
- **Kubernetes**: Kubernetes provides you with a framework to run distributed systems resiliently. Minikube was used for implementation of Kubernetes which created a VM on the local machine and deployed a simple cluster containing only one node. It takes care of scaling and failover for your application and provides deployment patterns. For example: Kubernetes can easily manage a canary deployment for the system.

- **Thymeleaf**: Thymeleaf is a modern server-side Java template engine for both web and standalone environments. Thymeleaf's main goal is to bring elegant natural templates to the development workflow - HTML that can be correctly displayed in browsers and also work as static prototypes, allowing for stronger collaboration in development teams.

System Overview

Main Components:

- AppController: The main controller of the application that has all the internal mapping to every API.
- FileUploadService: The service which is called when the user choses to upload his/her file to the database.
- FileDownloadService: The service which is called when the user choses to download his/her file from the database.
- QRCodeService: This service is used to generate the QR Code for our download page url.
- URLService: This service is used to encode a long url into a shorter url and also to decode it back to a long url.
- FileDeleteService: This service is called when the user chooses to delete any of his uploaded files in the current session.
- BaseConversionService: This service is called along with the URLService and is a helper service to it for encoding and decoding urls.
- AuditErrorService: This service is called whenever an exception occurs and is used to persist all the information about the particular exception.



Explain how your system works based on the diagram.

- The homepage/dashboard (“/”) initially shows an option to upload multiple files.
- Once files are uploaded they are displayed on this dashboard. Each file is saved to the database in the form of a byte array with a ‘longblob’ datatype. Other attributes of the file like name, size, upload date, session id are all persisted on our database.
- Each file has three services available for them namely view, delete, and download.
- Clicking on the name of the file will open the file in its intended format if the browser supports it.
- Clicking on delete will delete the file for that user session and will refresh the page without that file. This is accomplished by using an ‘active’ field for each file.
- Clicking on download will return the associated file row from the database and then the byte array content of the file will be written to the servlet output stream which allows the user to download the file in its original format.
- Clicking on ‘Generate QR and URL’ will call two services. First the download page url is constructed which will lead the user to a custom download page with all of their uploaded files. The user is provided all three option of file viewing, downloading, and deleting on the download page as well.
- This download URL is also used to generate a QR Code using the open source ZXing library. This QR code is displayed to the user and they can use a smartphone or any device where they want to view their file to scan the QR code and get the download URL. This was done as different devices even on the same network will have different session Ids and this was the way to share the download page url.
- There is also the scenario that the user wants to access his/her files on a different desktop or device that does not support QR code scanning. Hence the URLService uses the download page url to generate a shortened url so that it is easy to input into the device. This is achieved by a method similar to other online url shortening web apps. The actual url is saved in the database and a encoded url is returned to the user. When the user enters the encoded url the system again calls the URLService along with the BaseConversionService to decode it back into the original url.
- As the user can enter any url on the domain we have implemented a custom 404 error page that is triggered if the url entered by the user is not associated with us. This situation is triggered for other exceptions as well and we are persisting the details of such exception in the database. The user is also provided with a unique ID

for the error and in the event of an escalation the development team can use this ID to trace back the causes of the error and possible solution.

Explain how your system is designed to support scalability and fault tolerance.

Scalability:

In order to scale the application we have used kubernetes. Using this we provide more overall resources by running more instances. We have used multiple Pods, one for each instance. In Kubernetes, this is typically referred to as replication. Replicated Pods are usually created and managed as a group by a workload resource and its controller. A Replication controller makes sure that a pod or a homogeneous set of pods is always up and available.

Fault tolerance:

For the purposes of fault tolerance, we implemented a custom exception handling method that would help us catch and throw exceptions that occur eg. invalid URLs. This would help us avoid the bothersome Whitelabel error page and redirect our user to a custom error page with valuable information like error type, error description, and allow the user to navigate back home. We are also tracking all the exceptions that occur in our database with additional information like timestamps and the requested URL that caused the exception. We are also providing an error tracking ID that will help the development team track down the error in backend systems in the future.

Contributions

Provide a sub section for each team member that describes their contribution to the project. Descriptions should be short and to the point.

Josemon Baby:

Responsible for designing the system from the ground up using spring boot, Hibernate, MySQL database in a Spring MVC Controller pattern implementation. The initial approach was to develop a barebones system to upload and download but because of the requirement of not using unique user identifiers, had to use session Ids to track files by the same user. Designed the base template for the homepage and the dedicated download page. The file download feature was developed using large blobs as a datatype for byte array insertion. File deletion was also added in the form of setting an active or not attribute for a file. Developed a service to generate a shortened URL for a quality-of-life update for the users and also designed a QR code generator for the same. Shortened URLs also had a link

expiration feature to save database volume. I also created the base tables for tracking the files and shortened URLs.

Grishma Kagde:

Responsible for implementation of kubernetes and docker for scaling purposes. Minikube was used for implementation of Kubernetes which created a VM on the local machine and deployed a simple cluster containing only one node. Next ,a Kubernetes Deployment YAML file is created that specifies the configuration for a Deployment object, this is a Kubernetes object that can create and update a set of identical pods. Each pod runs specific containers, which are defined in the spec.template field of the YAML configuration. Also responsible for creating the Download and View Service which helps the user to download and view the uploaded file. Also contributed to the video and report of the project.

Yash Sowani:

Responsible for handling the exceptions that would assist us in recognizing and eliminating any errors, such as invalid URLs. Exception alerts are triggered when an error occurs which are sent to a monitoring system, ensuring that the exceptions are detected and addressed quickly. When an exception is caught, it can be logged to a database, along with information about the context in which the exception occurred. A writer Exception is thrown in the Spring Boot application which throws an error message and causes the error in the system. Also in charge of the system's full Frontend development, using html, css, Javascript, thymeleaf as a templating engine and bootstrap for simplifying styling.

Reflections

What were the key challenges you have faced in completing the project? How did you overcome them?

MiniO and AWS were initially going to be used for auto scaling. An AWS EC2 instance was used to launch our application. We intended to use Amazon EC2 Auto Scaling to automatically raise the number of Amazon EC2 instances during surges in demand in order to maintain performance and reduce expenses. However, the process took a long time for modifications to the user data script for the specific deployed EC2 instance. Finally, we made the choice to scale using Kubernetes. Although installing Minikube on the machine was a difficult undertaking, we eventually succeeded in running Kubernetes.

What would you have done differently if you could start again?

- Encoding of the original download URL is done for shortening purposes. This encoding is done right now using very straightforward and thus lacks the complexity of an actual cryptographic algorithm like AES. This would lead to a highly secure encryption and decryption methodology for our URL shortener.

- We had tried to use different microservice projects for each service but this turned out to be very complex and had to abandon the implementation.
- Hysterix could also be used for fault tolerance but we did not have the time to implement so we had to use custom exceptions.
- We used MySQL to store binary data and while it has its advantages, if the primary filetypes used in our applications are text, pictures, mp3 etc. a Filesystem might be a better choice. This difference will become apparent when we consider factors like fragmentation as storage age increases,fragmentation tends to increase[1]. This can be changed in the future depending on our user base. We can use Amazon RDS to have more scalability in terms of storage.
- We could have used React along with Springboot as that would help with the page redirects and refreshes impacting performance. This is because with react we can selectively choose only the component we need to re-render.

What have you learnt about the technologies you have used? Limitations? Benefits?

When using MySQL to store byte array data we had used longblob as the content datatype. This has some limitations:

- Pulling and pushing large BLOBs of data will clog up the memory cache is hosted database and will result in a performance hit.
- MySQL was designed as a monolithic single-node database and even though it works fine for small systems like ours, larger systems require the use of sharding or splitting a data set over multiple nodes.

That being said MySQL does provide faster and easier implementation onto springboot web apps and also provide lightning-fast lookups and data atomicity (in transactions).

During actions like upload, download, delete we are using page redirects of refreshing model view. While this method works as intended in our scenario, a large scale implementation of such a solution would have a significant impact on each action. This limitation can be solved by using solutions like react where only a component is re-rendered instead of the whole page. Also our download page using the session id parameter so it could be vulnerable to HQL injection attacks but this can be improved by using Criteria API.

The deletion service that is being used is achieved by using an ‘active’ field on the database. In future iterations actual deletion can be done by implementing a scheduler that runs everytime the application is deployed which removes all rows with an inactive state or which are associated with expired links. This can save us space.

References

[1] Gray, Jim. “To BLOB or Not to BLOB: Large Object Storage in a Database or a Filesystem.” *Www.microsoft.com*, 1 Apr. 2006, www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/.