Projeto Segurança Informática em Redes e Sistemas

THE CORK, GROUP A57

96902 | Pedro Gomes
100740 | João Salgueiro
105688 | Tiago Figueiredo

16 de dezembro de 2022

1. Infrastructure Report

I. BUSINESS CONTEXT

TheCork is a service that allows its users to taste the best food and wines in town by enabling easy restaurant reservations via a very user-friendly mobile application that integrates seamlessly with the restaurants' calendars.

TheCork offers a mobile app to its customers to display the list of restaurants available at the user's location. It also allows them to book a table for a specific number of people if the restaurant still has free tables. There is also a website that provides the same functionality.

TheCork also provides back-office for the restaurants which is used to manage the schedule and to approve or deny the customers reservations.

II. INFRASTRUCTURE OVERVIEW

The group decided to implement an infrastructure in OS Ubuntu v22.04. It consists of two VM's and one virtual network switch. VM CORK-DB has one network interface connected to the local network and works as a database server. CORK-WEB will work as a web application server being the service provider. CORK-WEB has two network interfaces, one configured for the local network, the other NATed to host computer network card. The switch emulates the network 192.168.0.0/24. CORK-WEB will also work as the firewall using Ubuntu built-in software firewall.

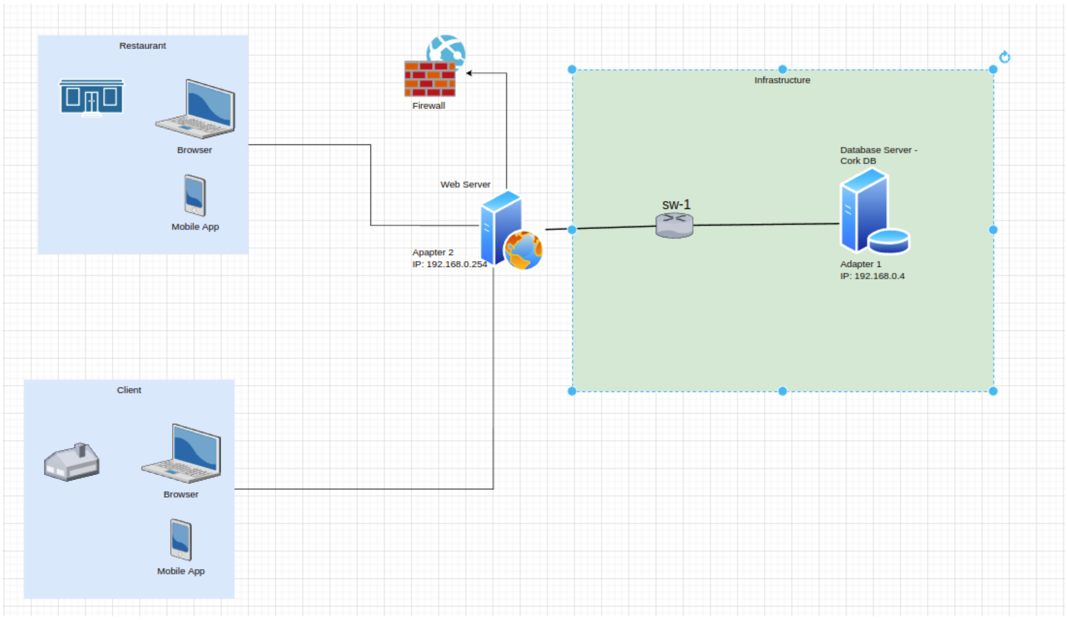
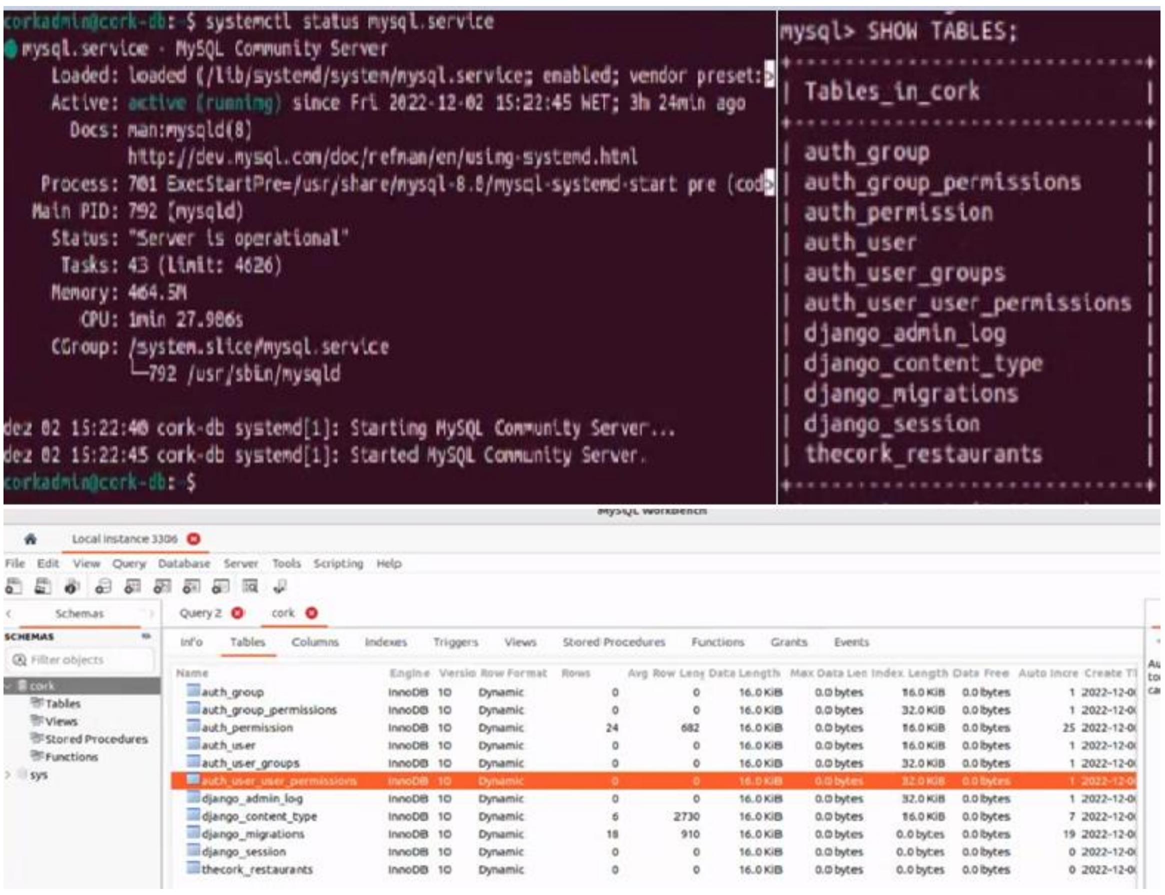


Figure 1 - Network diagram

For the database server, MySQL was installed, and an instance called cork was created, with it's own admin user "corkadmin". That instance will hold all the Tables for this project, as seen below.

To ease operation of the database server, the GUI MySQL Workbench was also installed.



```
corkadmin@cork-db:~$ systemctl status mysql.service
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:
     Active: active (running) since Fri 2022-12-02 15:22:45 WET; 3h 24min ago
       Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html
    Process: 701 ExecStartPre=/usr/share/mysql-8.0/mysql-systemd-start pre (cod
   Main PID: 792 (mysqld)
     Status: "Server is operational"
      Tasks: 43 (limit: 4626)
     Memory: 464.5M
        CPU: 1min 27.986s
     CGroup: /system.slice/mysql.service
             └─792 /usr/sbin/mysqld

dez 02 15:22:40 cork-db systemd[1]: Starting MySQL Community Server...
dez 02 15:22:45 cork-db systemd[1]: Started MySQL Community Server.
corkadmin@cork-db:~$
```

```
mysql> SHOW TABLES;
+----------------------------+
| Tables_in_cork             |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| thecork_restaurants        |
+----------------------------+
```

Name	Engine	Versio	Row Format	Rows	Avg Row Len	Data Length	Max Data Len	Index Length	Data Free	Auto Incre	Create Ti
auth_group	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	1	2022-12-0
auth_group_permissions	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	1	2022-12-0
auth_permission	InnoDB	10	Dynamic	24	682	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	25	2022-12-0
auth_user	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	1	2022-12-0
auth_user_groups	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	1	2022-12-0
auth_user_user_permissions	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	1	2022-12-0
django_admin_log	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	1	2022-12-0
django_content_type	InnoDB	10	Dynamic	6	2730	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	7	2022-12-0
django_migrations	InnoDB	10	Dynamic	18	910	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	19	2022-12-0
django_session	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2022-12-0
thecork_restaurants	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2022-12-0

MySQL Workbench
Local instance 3306
File Edit View Query Database Server Tools Scripting Help
Schemas
SCHEMAS
Filter objects
cork
Tables
Views
Stored Procedures
Functions
sys
Query 2
cork
Info Tables Columns Indexes Triggers Views Stored Procedures Functions Grants Events

Figure 2 – Database details

To implement the Web App, the group decided to use a framework called Django. That framework uses Python as a programming language and is structured on a design pattern called MVT (Model-View-Template) while providing a lot of custom built in administration tools for user management along with the multitude of Python libraries which could prove necessary for this project.

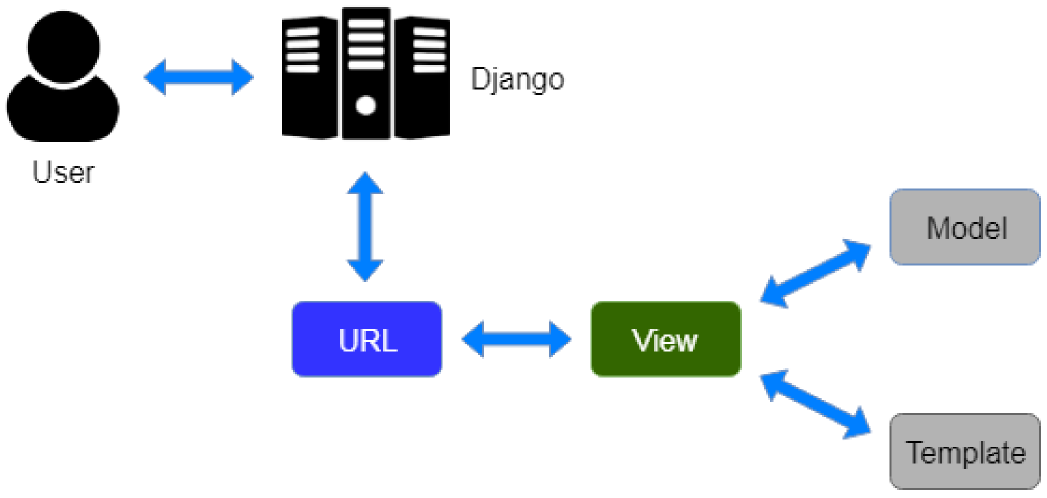
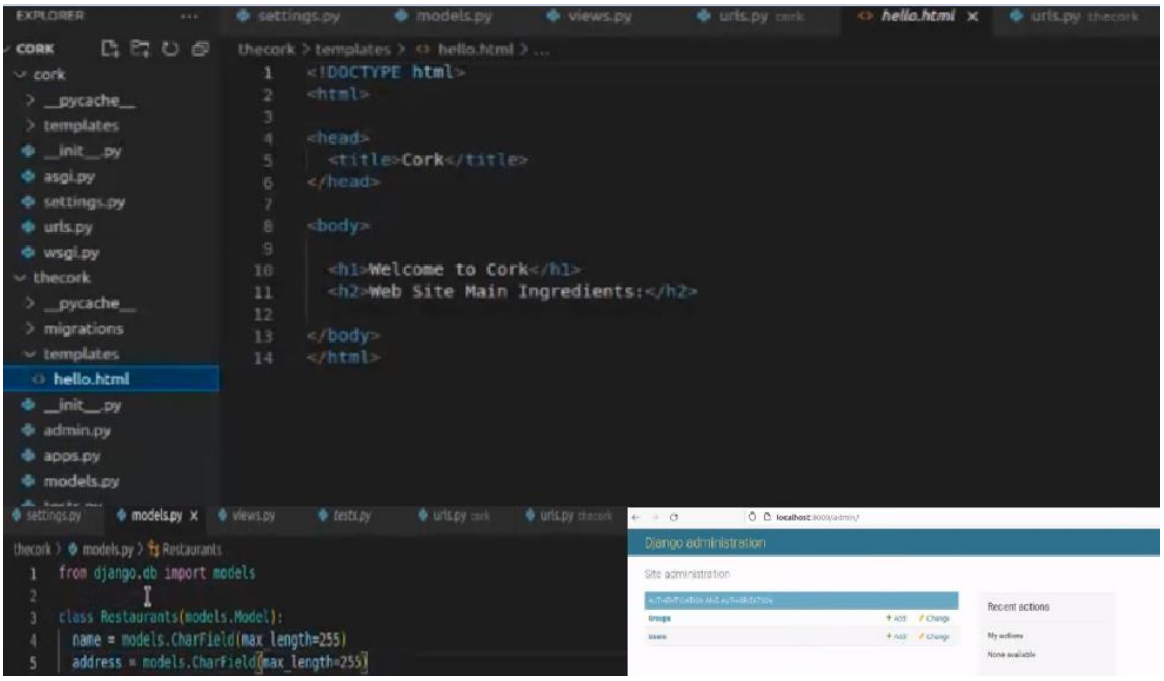


Figure 3 - MVT Pattern

The template part handles the user interface using html pages, the view executes all the logic and interacts with the model to carry data and the model handles the relational structure implemented in the database.



```
<!DOCTYPE html>
<html>

<head>
  <title>Cork</title>
</head>

<body>

  <h1>Welcome to Cork</h1>
  <h2>Web Site Main Ingredients:</h2>

</body>
</html>
```

thecork > models.py > Restaurants

```
from django.db import models

class Restaurants(models.Model):
  name = models.CharField(max_length=255)
  address = models.CharField(max_length=255)
```

Figure 4 - Django Overview

2. Channels Report

I. SECURE COMMUNICATIONS

The group plans to configure two secure channels.

1.Between the users and the WebServer, located in a DMZ, through a firewall (initially assumed a software firewall in the WebServer but a new infra-structure will be projected to add a VM server working exclusively as firewall and gateway).

2.Between the WebServer and the DatabaseServer in the Internal Network, through a Django connector that manages all database operations. This channel will also pass through the firewall (since it crosses Internal Network and DMZ).

Additionally, we plan on implementing a certificate authority (CA) instance on the WebServer (instead of creating a new server just for that role).

3. Security Challenge

I. CHOSEN SECURITY CHALLENGE

TheCork has personal information about each customer, which needs to be kept private. There is a need for an authentication server that can support both the app level and the back-office level, to avoid illegitimate users to get access to restaurant's agenda / customer data.

II. PROPOSED SOLUTION

Infrastructure

As exemplified below:

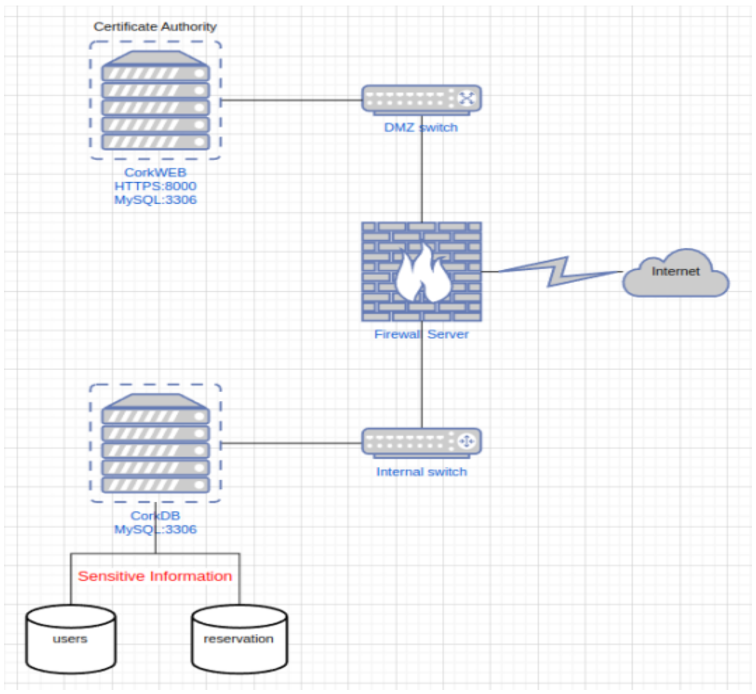


Figure 5 - Server Connections and Security Challenge

To ensure there are no connections using other protocols or ports other than the intended for the operation and management of the system the firewall must ensure:

All incoming HTTP requests (default port 80) must be forwarded to CORK-WEB. No other server in our infrastructure provides Web Services.

Only CORK-WEB can establish a connection with the database (on port 3306) using MySQL protocol.

External connections are only allowed to CORK-WEB.

Block unsecure or exploitable applications (ICMP, Netcat, etc.)

To ensure HTTPS access to the Web Service, we must provide users with a certificate. The website certificate will be self-signed (we are saying that our own server is secure, which is dubious in real world situations) but serves to exemplify how a secure connection works.

The CA server will provide users with certificates based on their public key and the CA private key. Our option is setting the Django server as a certificate authority, using python cryptography to encrypt data.

Application

A login function is already implemented that ensures confidentiality and restricts the functions that each user can access. There are two user groups: Owner (for restaurant owners, allowing CRUD operations over an Owner's Restaurants) and Client (for restaurant clients, which allow CRUD operations over a Client's Reservations). This information is stored in the database server in its own set of tables inside the application context.

One of the requisites is that no user can see other user reservations or personal information. This is guaranteed on the application side through:

-No unregistered access to the application is allowed.

-A client can only CRUD its own reservations and personal data.

-The owner of a restaurant can only CRUD his own restaurant's data and reservations made to them.

-Only the accounts marked as "staff" (currently only *corkadmin*) have access to all the information.

To prevent CSRF and session stealing we must secure the cookies for each user's session. Django also uses query parametrization, preventing SQL Injections to extract a user's data.

Currently all communication between the WebServer and the Database is over TLS v1.3 (transport layer security) to ensure data confidentiality and integrity.

References

Security in Django - https://docs.djangoproject.com/en/4.1/topics/security/

Django Databases - https://docs.djangoproject.com/en/4.1/ref/databases/#mysql-notes

MySQL Protocol - https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_PROTOCOL.html