

Scenario: BombAppetit

Group A24:

- Diogo Venâncio, 95555
- Pedro Martins, 99303
- Pedro Abreu, 92541

Business Scenario

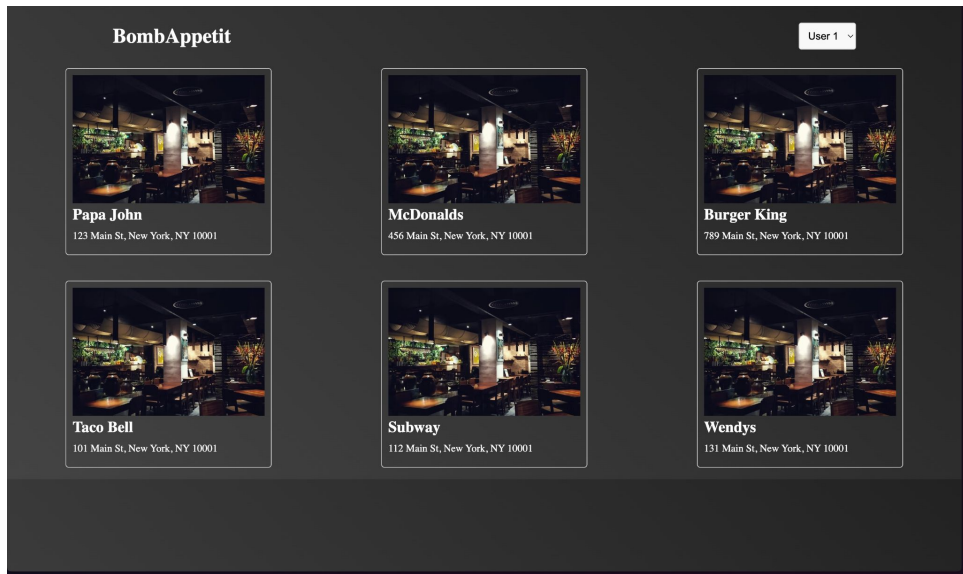
Web app for restaurant reservations

Services

- Web: SvelteKit
- Backend: FastAPI + Python
- DB: MySQL

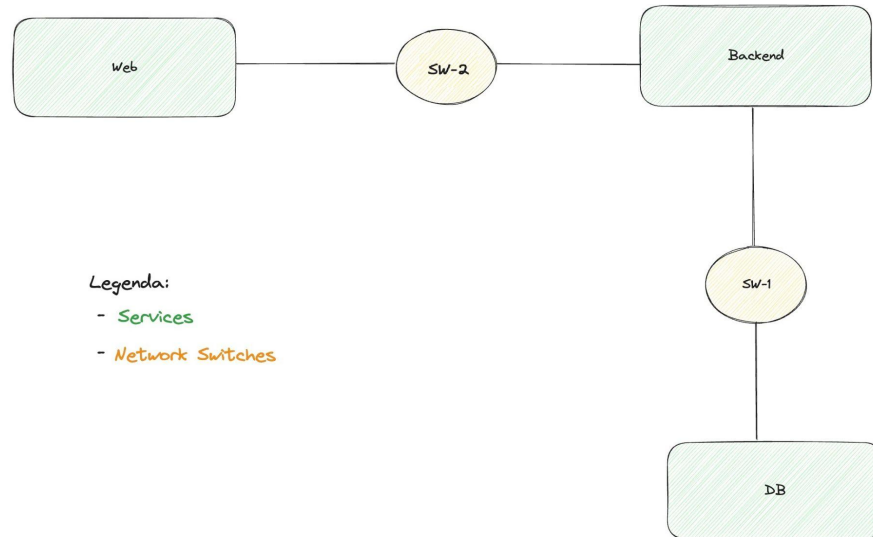
Security

- Network
- Vouchers
- Reviews



Infrastructure

- 3 VM's setup with Vagrant
- Isolated networks setup by Vagrant too
- Firewalls setup with iptables rules



Secure channels

- All communications encrypted with SSL
- Certificates and certificate keys generated on first run

```
if __name__ == "__main__":  
    uvicorn.run(  
        "app:app",  
        host=config.HOST,  
        port=config.PORT,  
        log_level="info",  
        ssl_keyfile="/home/vagrant/server-ssl/server-backend-key.pem",  
        ssl_certfile="/home/vagrant/server-ssl/server-backend-cert.pem",  
        ssl_ca_certs="/home/vagrant/server-ssl/ca-backend-cert.pem",  
        reload=True  
    )
```

backend

```
server: {  
    port: 3000,  
    fs: {  
        allow: ['/app/static']  
    },  
    https: {  
        key: fs.readFileSync('/home/vagrant/server-ssl/server-web-key.pem'),  
        cert: fs.readFileSync('/home/vagrant/server-ssl/server-web-cert.pem'),  
        //requestCert: true, ca: [ fs.readFileSync('/home/vagrant/server-ssl/ca-web-cert.pem') ]  
    },  
}
```

web

```
sudo tee -a /etc/mysql/mysql.conf.d/mysql.cnf > /dev/null <<EOF  
[mysqld]  
ssl-ca=/etc/mysql/ssl/ca-db-cert.pem  
ssl-cert=/etc/mysql/ssl/server-db-cert.pem  
ssl-key=/etc/mysql/ssl/server-db-key.pem  
require_secure_transport=ON  
EOF
```

db

Keys

The app uses elliptic curve prime 256k1

- Non-repudiation (ECDSA) + Key exchange (ECDH)
- Currently used in Bitcoin, Ethereum and Tor
- More secure and efficient than RSA

Web (users keys) and Backend store all public keys. The private keys are only stored in the servers they belongs to

No keys are transmitted over the network during application runtime!

Authenticity

```
def sign_data(data_bytes: bytes, server_private_key: ec.EllipticCurvePrivateKey, user_public_key: ec.EllipticCurvePublicKey) -> EncryptedData:
    data_hash = hashlib.sha256(data_bytes).hexdigest()
    encrypted_data = encrypt(server_private_key, user_public_key, data_hash.encode())
    return encrypted_data
```

- Our solution uses key exchange to encrypt the hash (SHA-256)
- Bonus: only the user requesting the data can successfully validate it
- Used to authenticate all HTTP responses' payloads backend -> web
- Meaning all restaurant, voucher, and review data (inside the payloads) is authenticated

Non-repudiation (security challenge)

- Key exchange provides authenticity, but doesn't provide non-repudiation
- When non-repudiation is required, the team used ECDSA

Vouchers

- When queried, the voucher data is encrypted using key exchange
- Only the backend and user assigned to the voucher can decrypt it
- Shared secret is used as key in AES-128 GCM mode
- Faster than asymmetric encryption, without sacrificing security

```
# Generate a symmetric key from public and private keys of both parties
derived_key = private_key_encryptor.exchange(ec.ECDH(), public_key_decryptor)

# Select algorithm and mode for symmetric encryption
iv = secrets.token_bytes(16)
cipher = Cipher(algorithms.AES(derived_key), modes.GCM(iv), backend=default_backend())
```

Security challenge: Gift voucher

- Key exchange is used to share the encrypted voucher data with the new owner
 - Uses a series of encryption and decryption steps
- The original owner signs the request and send the newly encrypted voucher data. The backend verifies and changes the voucher's user

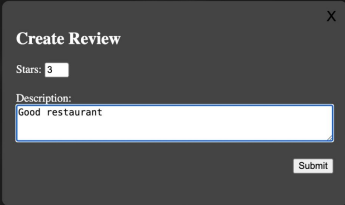
Reviews

Only introduced in the security challenge

To fulfil the **non-repudian** requirement, we added signature validation.

Flow

1. User A creates review and signs it
2. User B fetches review
3. User B validates signature with user A's public key

A screenshot of a 'Create Review' form. The form is dark gray with a title bar that says 'Create Review' and a close button 'X'. It contains a 'Stars' field with the value '3', a 'Description' label, and a text input field containing 'Good restaurant'. A 'Submit' button is at the bottom right.

Create Review

Stars: 3

Description:

Good restaurant

Submit

Demo

Conclusion

The requirements were satisfied and the team is proud of its work.

- Authenticity
 - All data
- Confidentiality
 - Vouchers
- Non-repudiation
 - Reviews
- Validation
 - Reviews