AAD / Backend Development

Security

Erik van der Arend / Jan Willem Boer

23/24 Q3

SAXION
UNIVERSITY OF
APPLIED SCIENCES

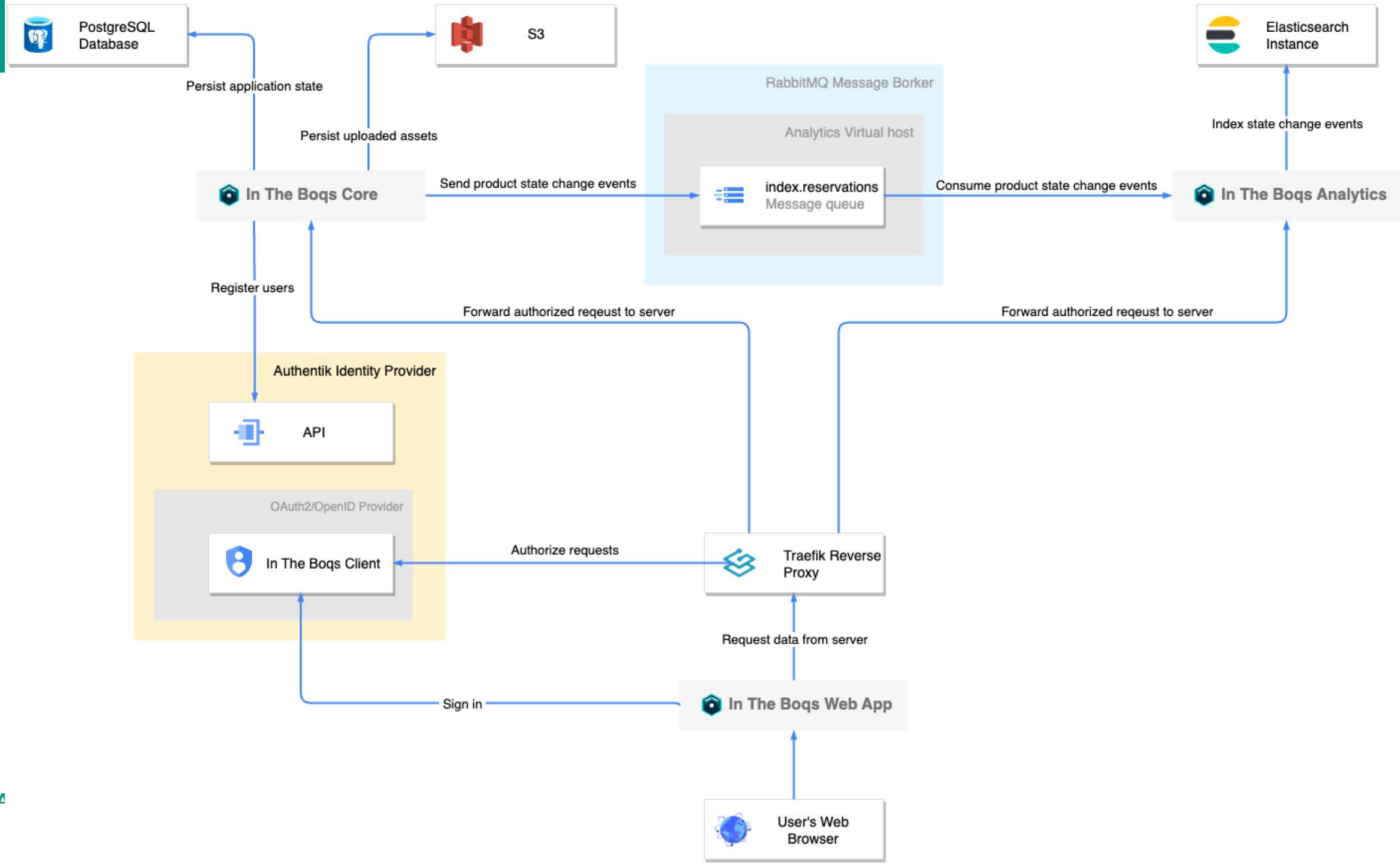
Assignment 2

The API itself must have:
- A self-hosted database
- A formal API
- At least 5 entities
- Code according to the arts of our trade
- Full dockerization
- Authentication with multiple roles
- Tests
- Advanced track stuff (depends)

Documentation

Documentation

- Context description and requirements
- Overview of the application and its components and its structure. For each part / component:
  - Technical details / diagrams
  - Choices made; alternatives considered
  - Libraries used
  - Subcomponents → repeat
- Deployment description
- Tests description / output
- Advanced track details


PostgreSQL Database
S3
Elasticsearch Instance
Persist application state
Persist uploaded assets
RabbitMQ Message Borker
Analytics Virtual host
Index state change events
In The Boqs Core
Send product state change events
index.reservations
Message queue
Consume product state change events
In The Boqs Analytics
Register users
Forward authorized reqeust to server
Forward authorized reqeust to server
Authentik Identity Provider
API
OAuth2/OpenID Provider
In The Boqs Client
Authorize requests
Traefik Reverse Proxy
Request data from server
Sign in
In The Boqs Web App
User's Web Browser

OWASP top 10

- Auth/auth is only the start
- OWASP keeps a list of security mistakes developer make

1 – Broken access control

An authenticated user can access objects that belong to a different user

https://www.example.com/account/10
https://www.example.com/account/1

How to prevent:
- Use random IDs
- Don’t include IDs in the URL if not needed.
- Couple data to specific account and check that.
- Deny by default: keep checking access at each call.
- Test!

2 – Cryptographic failure

Sensitive data is not properly encrypted “in rest and transit”.

Sending sensitive data as plain text over the line
Storing sensitive data as plain text in the db
Storing sensitive temporary data as plain text

How to prevent:
- Don’t store sensitive data
- If needed, store with proper encryption
- Use HTTPS (and similar)

3 – Injection

An application accepts data as input and processes it as instruction.

```
SELECT * FROM users WHERE id=$id
```

```
https://example.com/login?id=2
https://example.com/login?id=2 OR 1=1 LIMIT 1
```

```
SELECT * FROM users WHERE id=2 OR 1=1 LIMIT 1
```

- SQL injection
- javascript: inject script when submitting input to a website (XSS)
- Bash commands to a linux server

3 – Injection

An application accepts data as input and processes it as instruction.

Prevention:
- Use frameworks to inject data into templates
- Use sql parameters, never string concatenation
- Use templating frameworks for HTML

Other highlights

- Security misconfiguration: having default passwords, open ports, too verbose errors
- Outdated components (*cough* *wordpress* *cough*)
- Allowing programmatic high-rate access to your API

Further reading / watching

https://owasp.org/www-project-top-ten/

LinkedIn Learning summary of the OWASP top 10:
https://www.linkedin.com/learning/learning-the-owasp-top-10-9364599/

Now what

This week

- Continue with Assignment 2