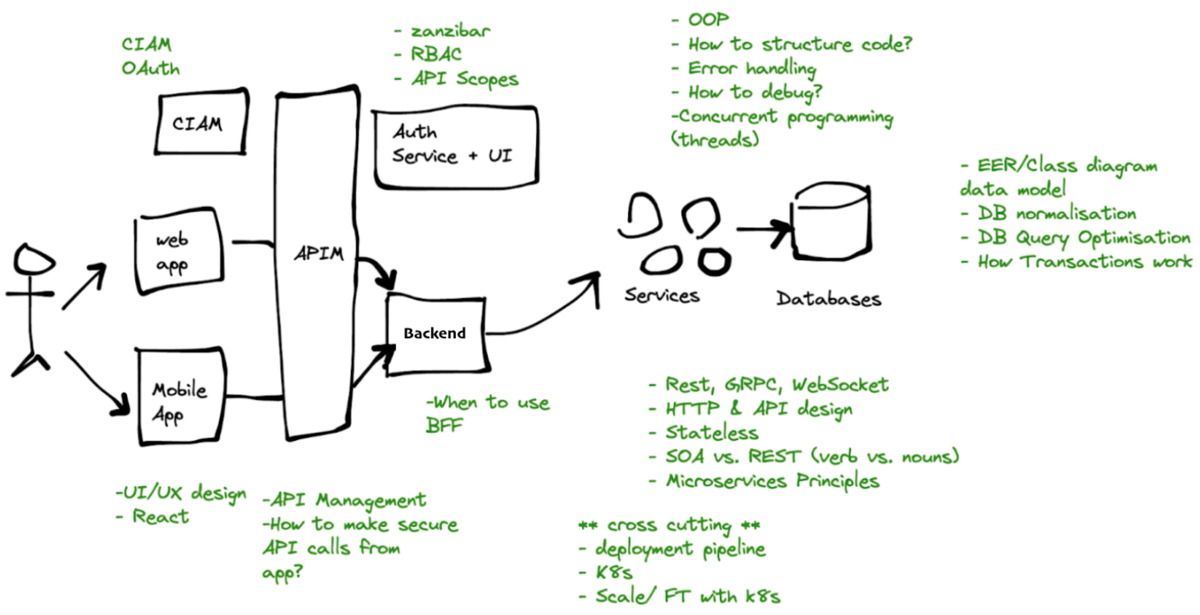
Note: This document is Work-in-Progress (WIP)

Author: Internal-Apps team

Overall structure

This structure is an example of how to deploy apps and different things to learn.
-Srinath



What to learn before you get started

Basics

- Refresh your OOP concepts, try to model a few domains using OOP and think through how those models will be used

- Structure code such that it can be read as a story. It is pretty easy to do when you understand it. Listen to next 5 mins in this talk Core Design Principles for Software Developers by Venkat Subramaniam

UML Diagrams

- Use Case Diagram: https://youtu.be/zid-MVo7M-E
- Entity Relationship(ER) Diagram:
    - https://youtu.be/QpdhBUYk7Kk
    - https://youtu.be/-CuY5ADwn24
- Class Diagram: https://youtu.be/UI6lqHOVHic

Databases:

- Database normalization: https://youtu.be/UrYLYV7WSHM

Domain Driven Design

- What is DDD - Eric Evans: https://youtu.be/pMuiVlnGqjk
- Article: Getting Started with Domain-Driven Design https://drive.google.com/file/d/1UrraLwLxjD9V_c6uxCdPzPOM7UZXh51d/view?usp=sharing

Cell Based Architecture

- https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md

API Design

- API should be intuitive
- Class, method & parameter names should give an indication of what the API does
- Document the API
- "If you write programs, you are an API designer"
- "When in doubt, leave it out"
- "Your API speaks back to you"
- How To Design A Good API and Why it Matters - By Joshua Bloch
- Check out the APIs written by GitHub - https://docs.github.com/en/rest?apiVersion=2022-11-28

Code Quality

- Get your architecture, code & UI reviewed
- Make sure to send pull requests feature-wise, with at least one PR every two days, instead of sending the entire implementation at once. Also it would be convenient for the PR reviewer.
- Follow coding best practices
- Follow framework conventions
- Don't over-engineer!
- Reading code should be like reading a novel!
  - e.g. makeTea()
- Write beautiful code!
- Remember, it is your reputation that is out there!

Committing Code

- Request (read-only) access to wso2-enterprise repositories by filling the Git Committers form.
  - Product team: internal-apps
  - Organization: wso2-enterprise
  - Repo names: digiops-hr, digiops-sales, digiops-finance, digiops-infra, digiops-marketing, digiops-engineering, digiops-superapp, Internal-app-product-management
  - Access: read-only

- Git and GitHub for Beginners - Crash Course
- How to Write a Git Commit Message https://cbea.ms/git-commit/
- Some of the best Git/GitHub practices
  - GitHub - Secure your Account with Two-Factor
  - Configure the correct email address and link to your GitHub user
  - Write meaningful commit messages
  - Don't commit generated files
  - Make Effective use of Branching
  - Use .gitignore to Prevent Tracking Files (Also, create a meaningful git ignore file)
  - Don’t leak secrets into source control
  - Don’t commit dependencies to the source control
  - Don’t commit local config files to the source control
  - Use the correct naming convention for repos/branches/tags

GraphQL

- Getting started with GraphQL: https://graphql.org/graphql-js/

- Ballerina GraphQL: https://lib.ballerina.io/ballerina/graphql/latest

REST, HTTP

Learn the basic REST principles

  - http://www.infoq.com/articles/webber-rest-workflow
  - Pay special attention to how HTTP is used
  - Understand what media types and content negotiation means

WSO2 Rest API Design Guidelines

- https://wso2.com/whitepapers/wso2-rest-apis-design-guidelines/

Ballerina
- https://learn-ballerina.github.io/index.html

Which HTTP methods match up to which CRUD methods?

- https://stackoverflow.com/questions/6203231/which-http-methods-match-up-to-which-crud-methods

Note: After you design the APIs, remember to go through the guidelines again and use them as a checklist.

Learn HTTP

  - http://www.jmarshall.com/easy/http/.
  - You should be able to Understand
    - HTTP verbs
    - Status codes
    - Headers

REST vs GraphQL

- Rest vs GraphQL: https://youtu.be/PeAOEAmR0D0
- Vs WebSocket vs gRPC

Identity Concepts

- Understand the difference between Authentication vs Authorization
- Learn RBAC
- CIAM - How to use it at webapp/ mobile app level and API security level

OAuth 2.0

- OAuth 2 basics: https://wso2.com/blogs/thesource/2019/08/oauth-2-basics/
- Grant types and when to use them (authorization_code, password, client_credential, refresh_token)
- Client types and when/how to use them (Public client, Non-Public client)
- Other common terms and meanings (example: well-known endpoint etc)
- SSO with OAuth 2.0

Concepts involved when using OAuth 2.0 Application/Service Provider

- Client_id
- Client_secret
- Grant_type
- Redirect URL
- Token types (OAuth access token types and how to validate them)
- Understand the token exchange flow

Get a basic knowledge of JWT (Generation, Validation: validate with the certificate, validate with jwks, etc)

K8s - Constructs, Ingress

//TODO: Add learning resources

Event-driven architecture, Messaging framework

//TODO: Add learning resources

What to learn to get started

Application Security

This is the 1st task you have to attend when starting a project.

- Secure Software Development Training 2024
    - Master: Structured Training and Regularization for Intrinsic Diverse Embedding Language Models(STRIDE-LM)
- Threat Model - Security Advisory Automation
- Application Deployment Security Checklist

User Ballerina Authorization Module.

Ballerina

Get started:

- https://ballerina.io/learn/get-started/

Best practices

- https://learn-ballerina.github.io/
- https://github.com/ballerina-platform/ballerina-lang/blob/master/CONTRIBUTING.md#submit-your-contribution

Go through some of the samples

- https://ballerina.io/learn/by-example

To learn more about particular sample, go through the spec

- https://ballerina.io/learn/by-example/http-error-handling/
- https://github.com/ballerina-platform/module-ballerina-http/blob/master/docs/spec/spec.md#82-error-handling

Ballerina testing

- https://youtu.be/13J_WnJooAA?si=v5XTtzDBKLyMQtDj

Asgardeo

Get started:

- https://wso2.com/asgardeo/docs/get-started/
- Try a sample app: https://wso2.com/asgardeo/docs/get-started/try-samples/

Choreo

Introduction to Choreo Service Components - 2023-07

How to Create a Service in WSO2 Choreo | Choreo Community Call #1

How we Use Choreo inside WSO2

- WSO2's Digital Transformation Journey with Choreo: A Platformless Approach | W…

Get started:

- https://wso2.com/choreo/docs/get-started/what-is-choreo/

Bring Your Own Repository

- https://wso2.com/choreo/docs/tutorials/connect-your-own-github-repository-to-choreo/

Integrating Ballerina, Asgardeo, and Choreo: A Sample Project

- https://medium.com/@jishanrandika/be-a-smart-developer-with-ballerina-choreo-and-asgardeo-fd8db3c3516b
- https://medium.com/@rashmika-silva/effortless-abc-building-secure-scalable-apps-with-asgardeo-ballerina-and-choreo-b818e84bca95

Points to remember

When using external libraries, icons, fonts..etc.

Please get permission from your mentor before using any.

- We can use google fonts, but We CAN'T use google **hosted** fonts. So for going forward download required google fonts and use them.(you can ignore the font files from .gitignore). So when building your frontend, the developer needs to download and place the font in the proper place etc. MUI is also using google fonts but the theme configuration **has an option to use self hosted fonts, we have to use that option**. (to confirm where your fonts are downloaded, you can inspect your app and go to the sources tab, which will show all the sources where fonts are downloaded)

Use internal channels to communicate;

- Internal Apps team email:
    - Subscribe internal-apps-group@wso2.com to receive emails.
    - Create an email filter for the above group as well (it will help you to easily find the emails received to our team).
    - Communicate/Discuss your project updates (at least weekly) through addressing the same group. This is a public email group visible to everyone in WSO2 (if subscribed).
    - Don't mix it up with the internal-apps-team@wso2.com which is a private email group for sensitive discussions.
    - Contribute to other discussions.
- Communicate through google chat group to seek help or start a discussion.

Use wso2-internal channels to communicate;

- WSO2 Channels
    - WSO2 Discord server: https://discord.gg/Xa5VubmThw
    - IAM - wso2is.slack.com
    - APIM - wso2-apim.slack.com
    - Ballerina -
        - https://discord.com/invite/wAJYFbMrG2
        - ballerina-platform.slack.com
    - Choreo - wso2-choreo.slack.com
- Communicating with other teams
    - Make sure to pick and choose the right email based on subject and context.
- Subscribed to the relevant mail threads using Email group manager app

Collaboration and Ownership

- Always discuss your ideas on the relevant mailing lists. There is no such thing as "over-communication" at WSO2
- Meritocracy http://theapacheway.com/merit
- Reuse, don't reinvent!
- Don't try to solve all the problems on your own
- If you are stuck, shout for help! Don't suffer in silence!
- If you see a problem fix it, at least let everybody else know

Your progress;

- Discuss with your team lead/mentor about the project and relevant stakeholders
- Prepare an initial SRS
- Define a milestone plan
    - Agreed-upon chunk of your work completed to a certain standard.
    - You can have a sprint based (typically two weeks) as well as feature wise milestones according to the project.
- A Milestone should include
    - Design reviews
    - Code reviews
    - Automation tests
- Milestones
    - Objective related milestones – These support the ability to objectively evaluate progress towards the overall objectives of the team.
    - Fixed-date milestones – Not everything, however, Implementation of apps and integrations also relies on external events, third-party deliverables, and external constraints. These are often fixed-date milestones that are distinct from the development dates.
    - Tasks should be sorted according to priority, and grouped by feature or release.
    - These iterations need to be mirrored in GitHub with Milestones.

- Fill in your progress
    - https://github.com/orgs/wso2-enterprise/projects/102

Releases

How version numbers are assigned and incremented https://semver.org/

Best Practices

Choreo Component Versioning

- Always create the first version (v1.0.0) from the main branch and use it for deployments.

- Avoid creating extra deployment tracks unless there's a specific use case for a separate version.

Choreo Project naming convention

- By Domain name (Finance, Sales)
  - Mainly entity services, business services, integrations and cron jobs
- By Super App name (HRIS Web, EMA Web) [**Web** suffix will differentiate it from others]
  - Web applications and backends

Choreo Component naming convention

- Web Applications: <Application Name> (Travel, Allocation, Promotion, MIS, Employee)
- Backend of the web apps: <Application Name> **Backend**
  - E.g. EMA Credit Card Backend
  - E.g. Promotion App Backend

Ideally a **backend service/component** only can have **ONE consumer** (This is how we could differentiate a backend vs business service.

- Business services (Integration services)
  - **How to identify a business Service:** services which is consumed by multiple clients or has a potential to be use by multiple clients in the future
    - E.g. Email alert **Service,** Salesforce and Netsuite sync **Service**
- Cron jobs or scheduled tasks (No need to maintain a suffix as Choreo lists the type of the component and the crons are not visible in dev portal as well)
  - People HR Sync
  - Netsuite Sync
  - Fetch office bookings

Web app template to get you started

  - https://github.com/wso2-enterprise/Internal-app-product-management/tree/main/webapp-template