Playbook Methods Repository

# **Back-End Implementation**

Implement services for business logic and data handling to support devices or a front-end implementation.

### Remote Agility: **•** High

### Linked Tactic(s): Agile Development, Solution Architecture

## Why we do it:

Back-End Implementation is the development of server-side, or cloud-based, business logic that powers a website, app, IoT device, or other service with centralized data storage and processing needs. Common back-end services include interacting with databases, file storage, 3rd party API integrations, and authentication and authorization.

This method typically includes building an API, which acts as a common means of communication with all the server-side resources, facilitating the necessary data for an app.

## 

## When to apply it:

In Agile Development and Solution Architecture:

* Centralized Computing: A back-end is typically employed when there is a need to centralize the storage and/or processing of data - e.g. files, or databases. This need may arise because users and their data interact, e.g. sharing photos via social media, or sharing information across an organization. In other cases, an organization may wish to “own” their data and so collect it in a central location.
* Facilitation of communication: Some services may not store data, but act as a means through which users, devices, or systems may communicate. E.g. an instant messaging app whose messages are stored only on the users’ devices to facilitate security and privacy.
* APIs and Services: Not all software has a Graphical User Interface (GUI). Some software may have an Application Programming Interface (API), an interface through which other programs may communicate. These services offer pre-made solutions to save application developers time and effort for many common functions, e.g. authentication, file storage (AWS S3.)

## Best Practices & Considerations:

* Test-Driven Development
  + Facilitates automated testing and deployment (see CI/CD)
  + Improves reliability of back-end software
  + Encourages well architected software increasing agility by making software easier to understand, change, and extend
* CI/CD
  + Back-end systems typically involve the coordination of many infrastructure components. Continuous Deployment lowers the risk and increases resilience of deploying updates to back-end services.
* Pair Programming
  + Facilitates Continuous Integration by eliminating the need for branching and code review
  + Improves reliability
  + Facilitates knowledge sharing and cross-team collaboration

## Responsible roles:

* Software Engineers: create the tests (see [Test-Driven Development](https://docs.google.com/document/d/1OT82TR7HbvPPfTkn-Ds37QaULahPy5xhk459lylmOPk/edit)) and software that runs on the server, making up the back-end.
* QA Engineers: apply the [Test Automation](https://docs.google.com/document/d/1HKQP85Qj1gfh9CEhI4To063Hw1pDQlSReSMALOU4Q7A/edit#) method to the back-end software and infrastructure.

## Tools:

* Online tools/platforms/services
  + Cloud: AWS, Azure, GCP
  + C# ASP.net
  + Java Spring Boot
  + Nodejs/Express
  + LAMP Stack
* Databases
  + SQL: PostgreSQL, MySQL, MSSQL, Oracle
  + NoSQL: MongoDB, CouchDB, Cassandra

## 

## Thoughtworks Examples - Linked

* Client working docs, airtable, miro/mural boards
  + xx
* Client polished presentations/deliverables
  + xx
* Internal assets - clinic materials / guild docs
  + xx

## 

## Learn more: How we do this?

* Templates (docs, decks, sheets, miro, etc.)
  + xx
* How-To Resources (external or internal)
  + xx
* Outside References (articles, books, etc.)
  + xx
* Sub-set Activities
  + xx

## 