Strategies for Resilient Web Applications and Websites

Zach Watkins
Software Applications Developer III
Communications

Things to keep in mind

- Resilience is failure recovery capability.
- Adding resilience is a continuous process.
- Impact determines priority.
- Available resources also determine priority.
- This feature takes time and requires planning.

Texas A&M Transportation Institute

Case Study

Production IoT Sensor Data Application

The Bicycle and Pedestrian Data Exchange

The Texas Bicycle and Pedestrian Count Exchange (BP|CX) is the official data repository for TxDOT's nonmotorized count data.

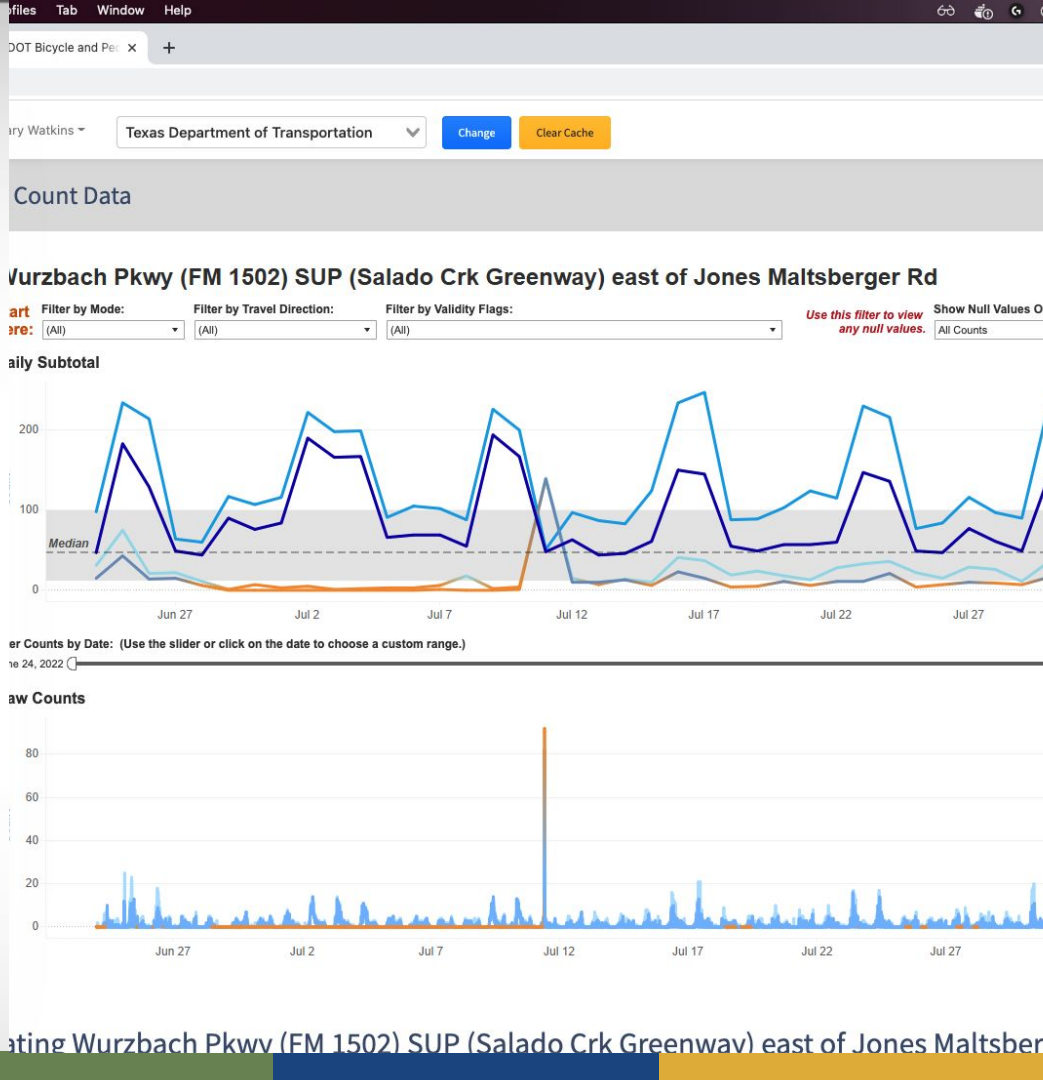
The system was born out of a need to collect, quality review, factor, export, and display data of many different types collected by different organizations.

The BPCX has quickly become the national leader in managing nonmotorized data; no other system has the features or abilities all wrapped into one package.

TxDOT is continuing to support the development of the BPCX to promote the use and integration of these data into its planning and construction efforts. This ultimately will improve bicycle and pedestrian facilities throughout the state and have a measurable impact on transportation equity and the environment.

Why resilience is needed

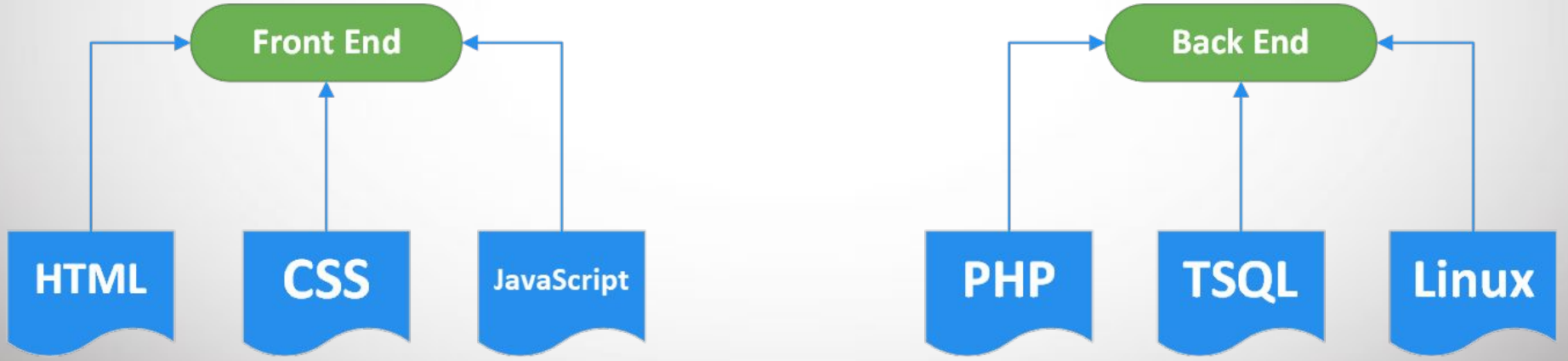
- ~66,000,000 rows of validated, quality-controlled data
- > 450 installations across Texas
- Used by 20 state agencies
- One developer (soon to have 2)



BP|CX

Technology Stack

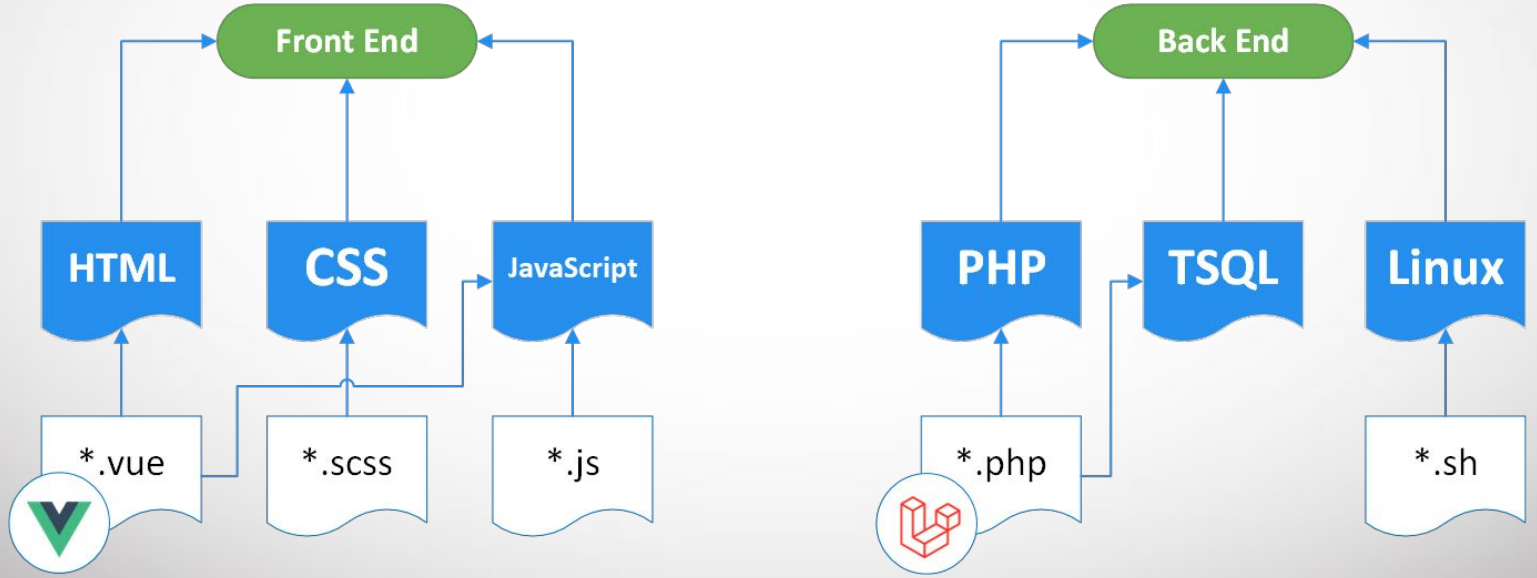
(runtime languages)



BP|CX

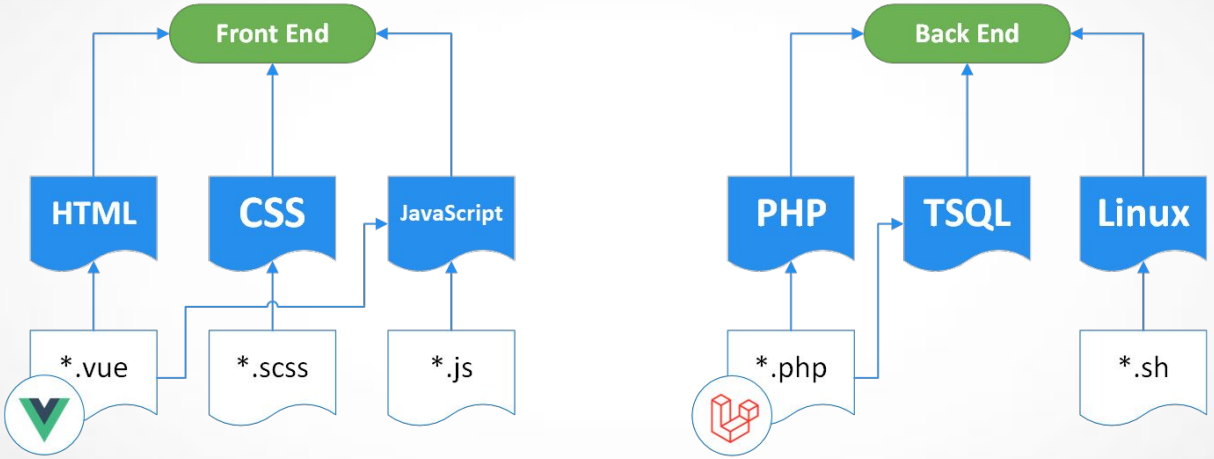
Technology Stack

(development)



BP|CX

Technology Stack
(development)



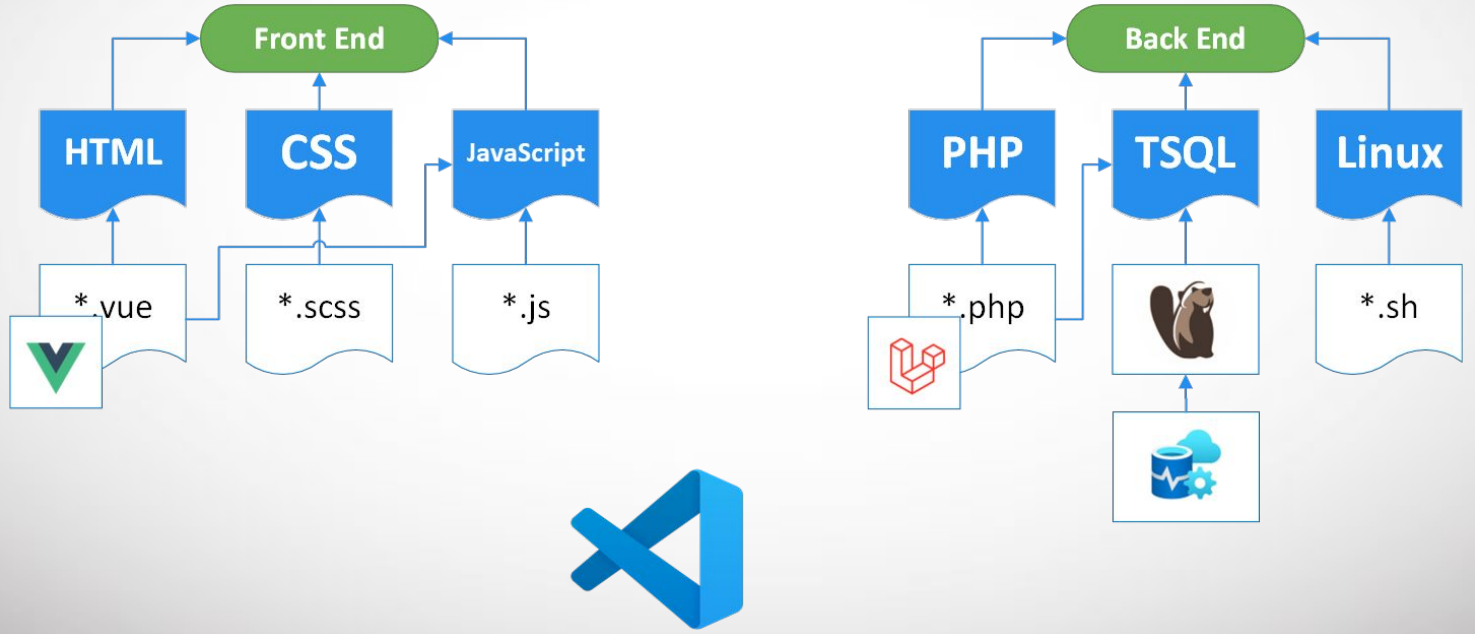
Integrated Development Environments

DBeaver

BP|CX

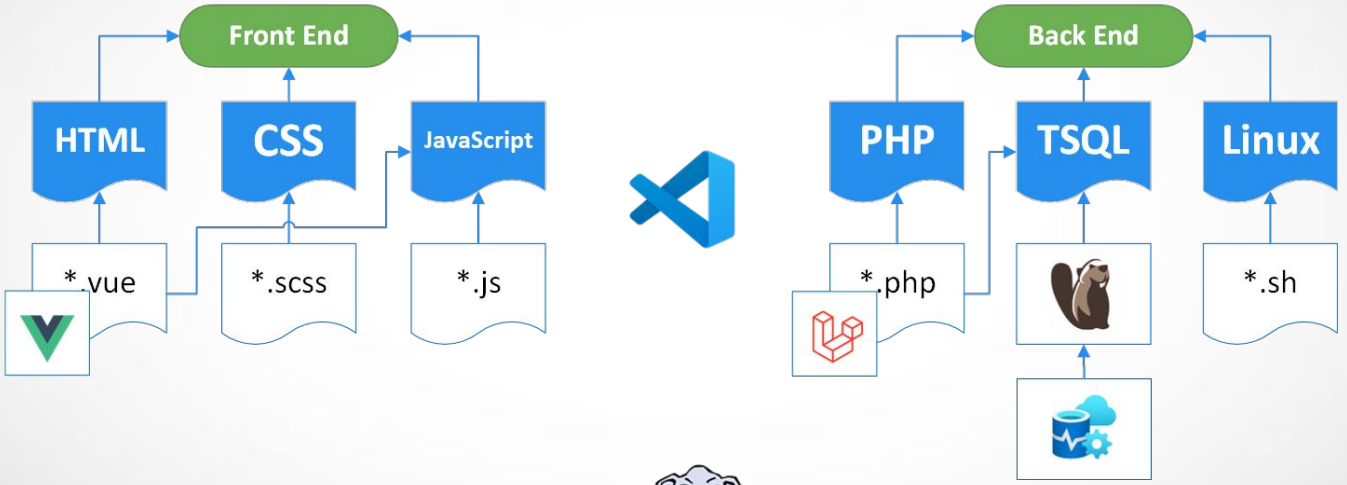
Technology Stack

(development)



BP|CX

Technology Stack
(development)

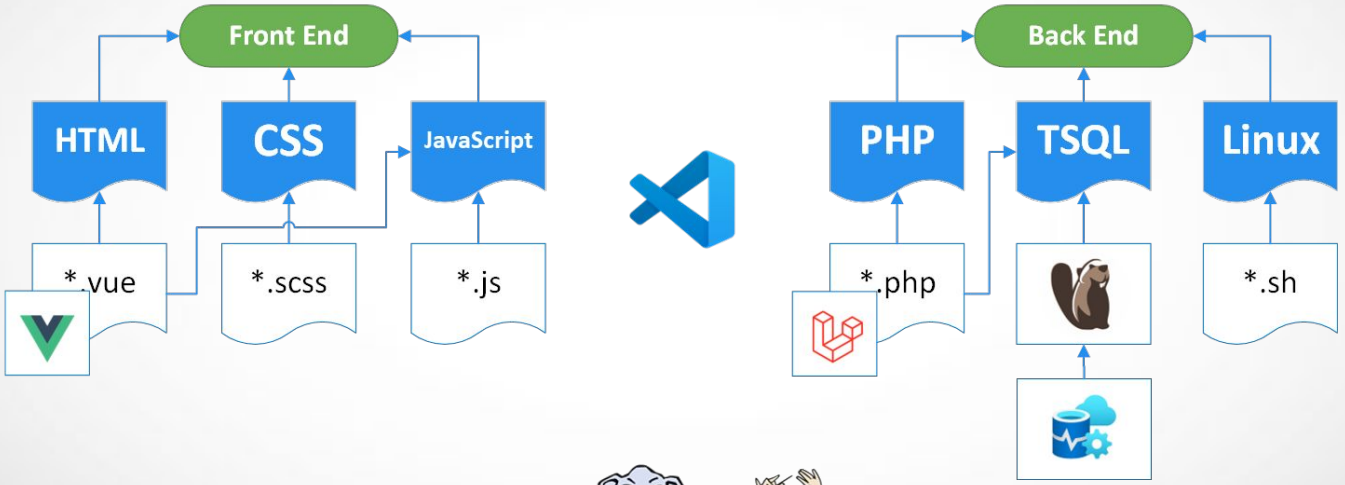


npm GitHub

Software Supply Chain

BP|CX

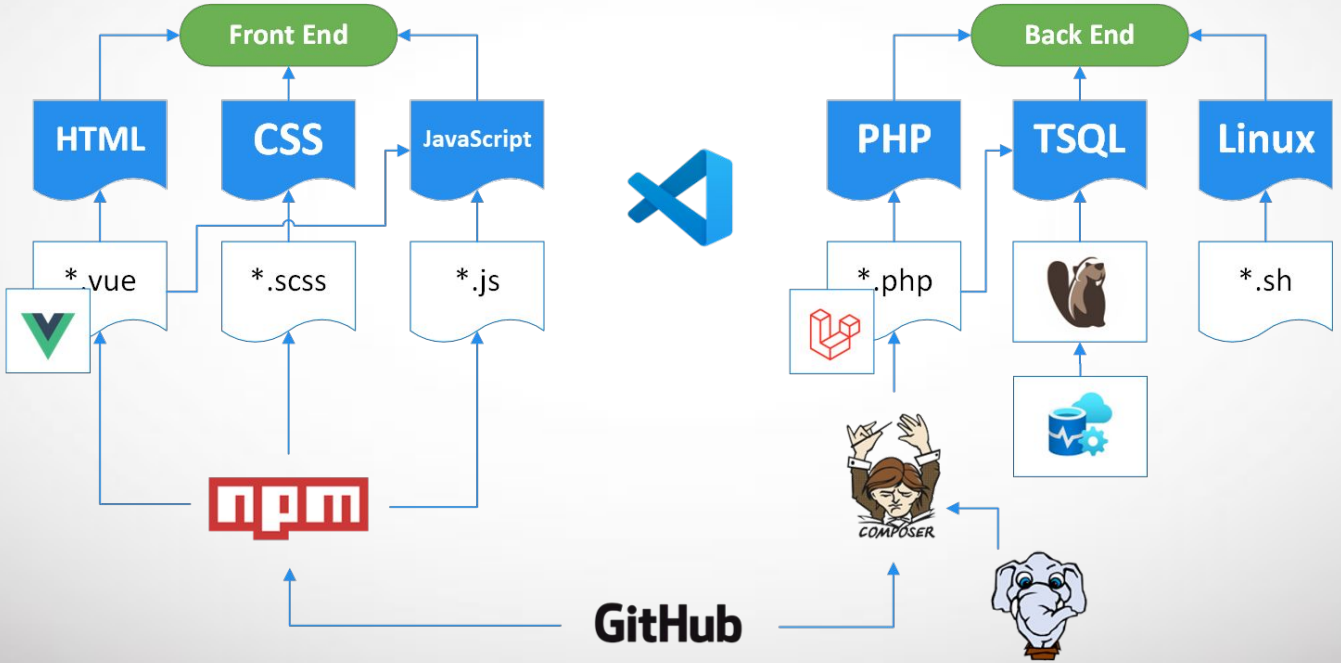
Technology Stack
(development)



Software Supply Chain

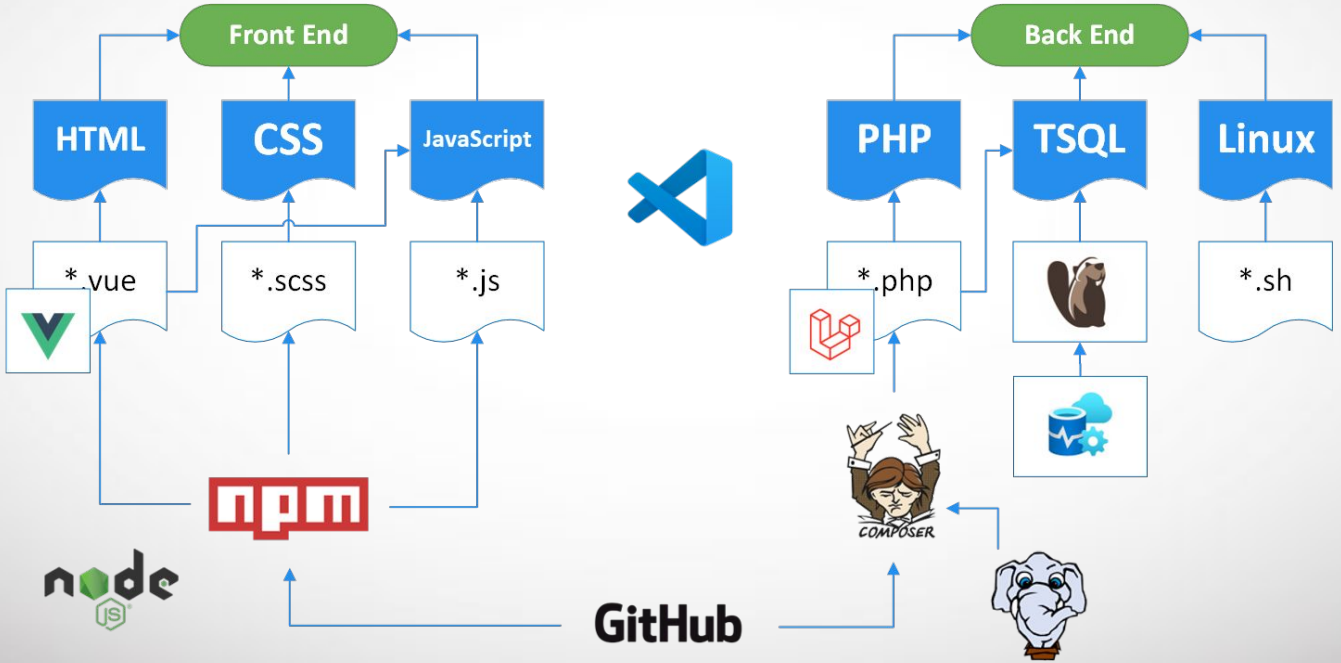
BP|CX

Technology Stack
(development)



BP|CX

Technology Stack
(development)



BP|CX

redis

docker

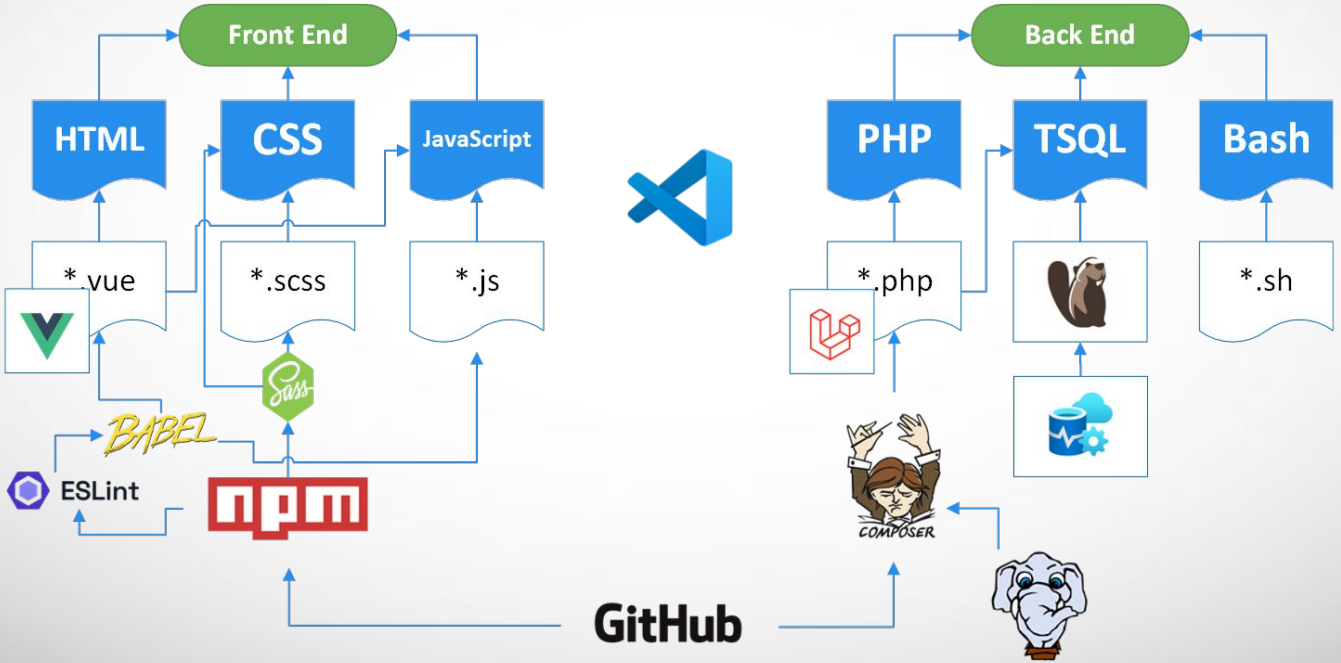
php

Ubuntu

SQL

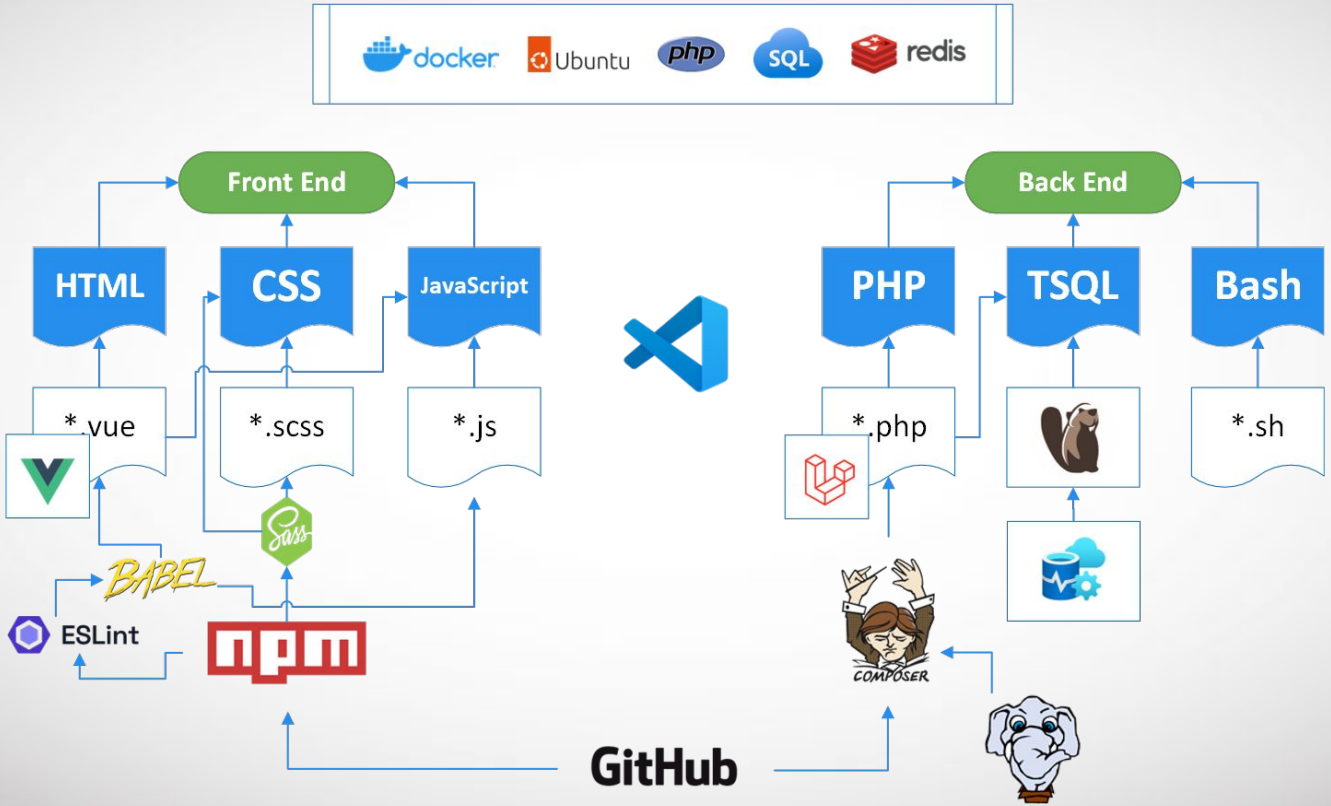
Technology Stack

(development)

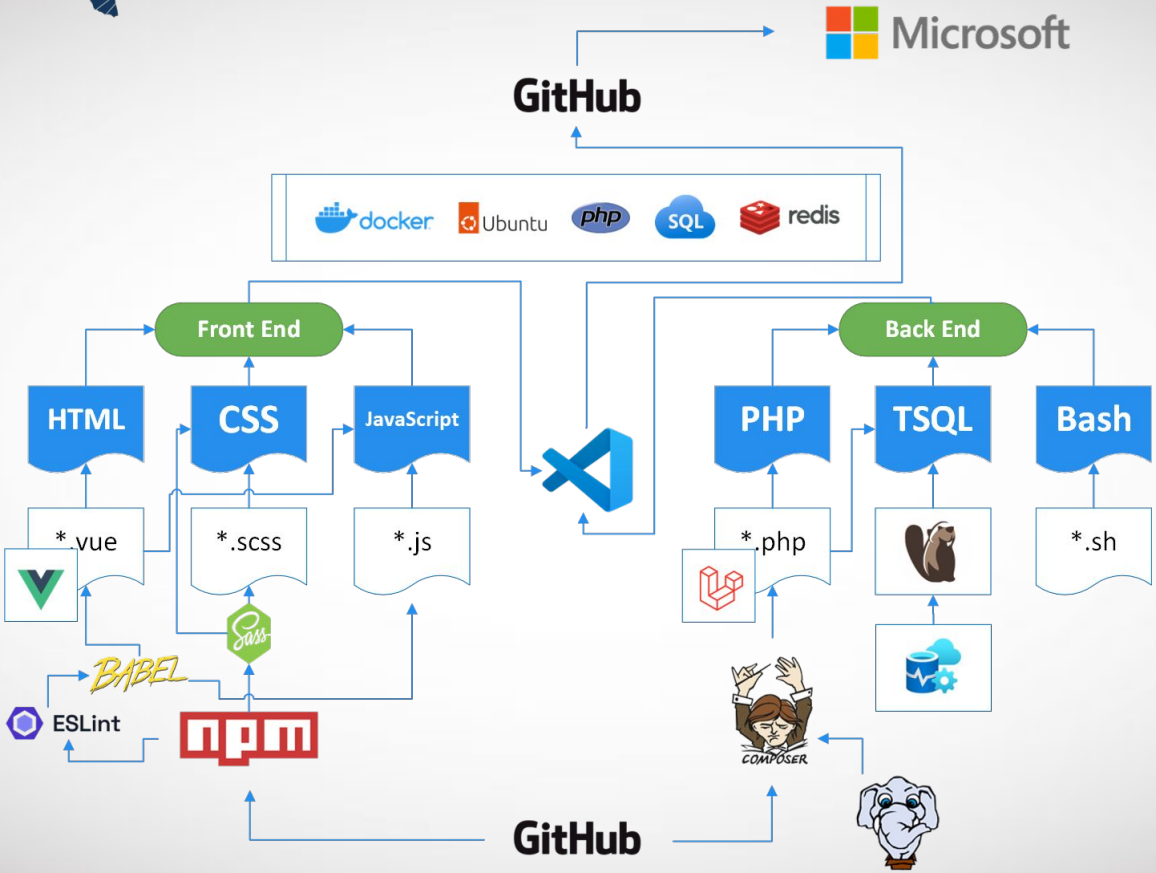


BP|CX

Technology Stack
(development)

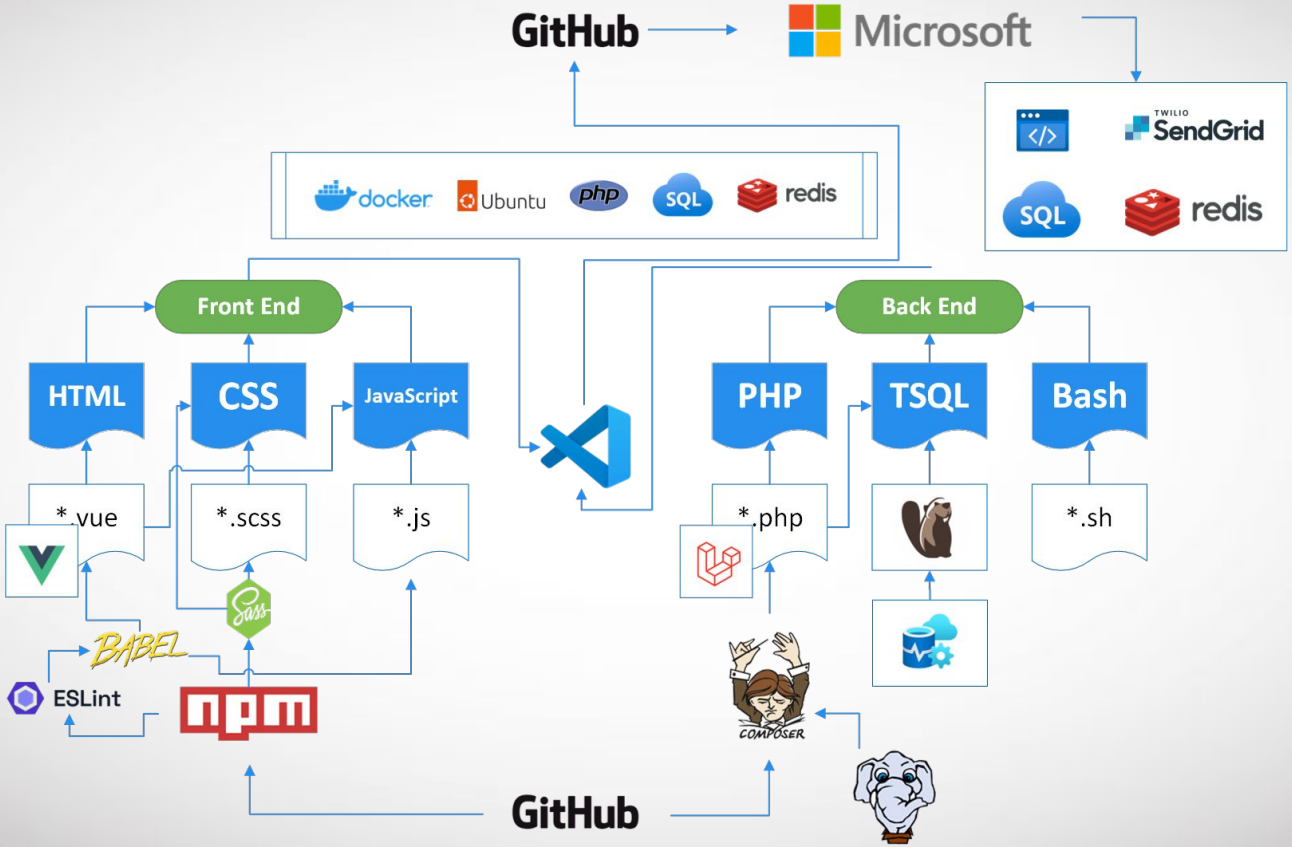


Technology Stack
(development + distribution)



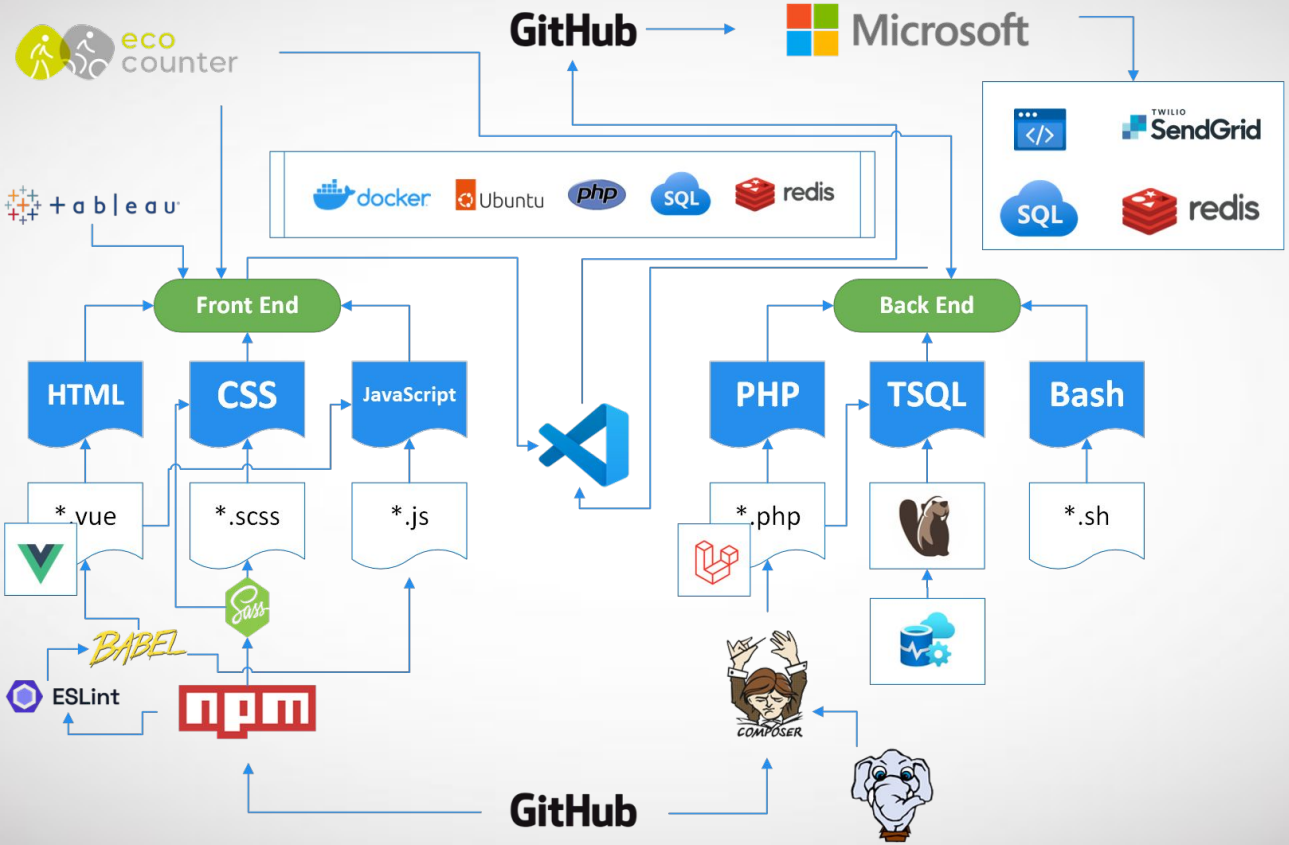
BP|CX

Technology Stack
(development + distribution)



Technology Stack

(development, distribution, & third party APIs)



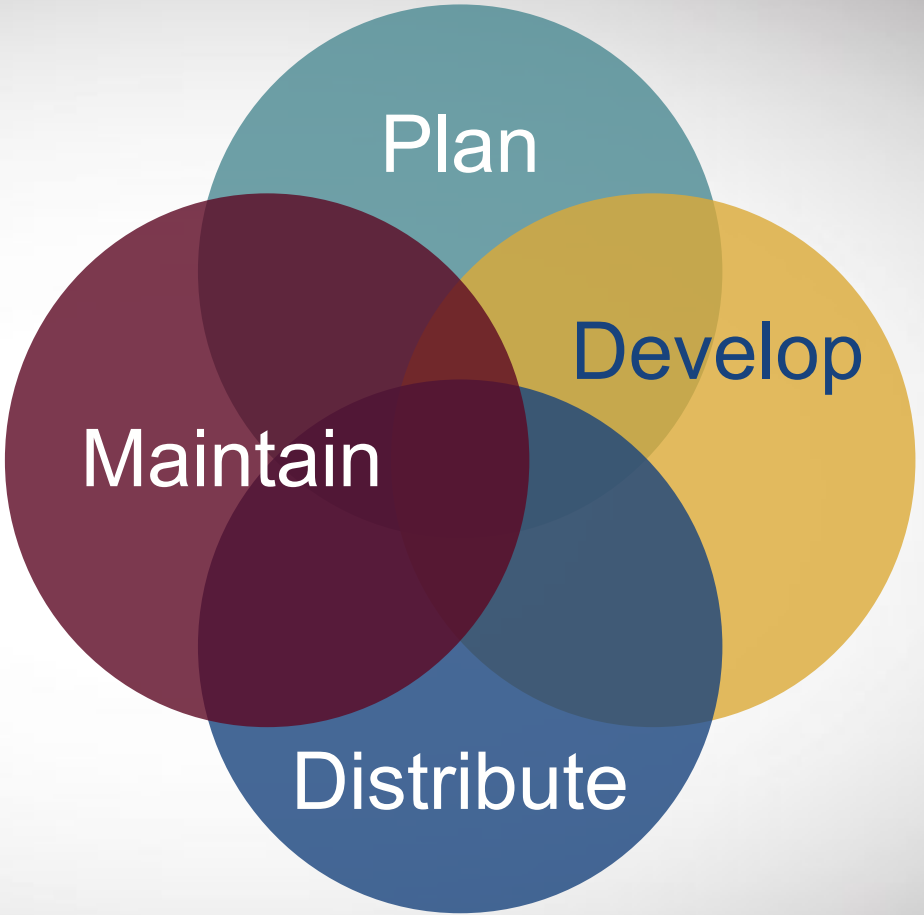
Software Development Cycle

A brief summary and not an official definition.

There's some overlap

It's a continuous, iterative process.

Response time is critical.



Planning Phase

- Consider potential points of failure like user error and connected services.
- If you use services they expect you to plan for availability issues:

  https://learn.microsoft.com/en-us/azure/azure-sql/database/develop-overview?view=azuresql#resiliency
- A useful error message is an acceptable solution.
- Consider notification messages in your designs as a viable communication channel for certain issues.

Development Phase

- Consider relevant software design patterns.
- Create scripts for your development steps for team resiliency.
- Document as much as you can - your team will thank you.
- Writing test scripts takes time; so does manual testing.
- Log application activity if it improves your incident response time or helps you improve your software.
- Decouple business logic from frameworks and third party services.
- Use staging environment for stakeholder feedback, dev environment for service testing, local environment for rapid development.

Retry pattern

- If a service is unavailable, retry the attempt - it's likely a temporary problem.
- Exponential backoff (waiting longer and longer between retries) prevents a service from being overwhelmed.
- Jitter randomizes a modification of the backoff duration.
- Try to keep a list of known transient connection error codes and only retry when they are encountered.

Circuit breaker pattern

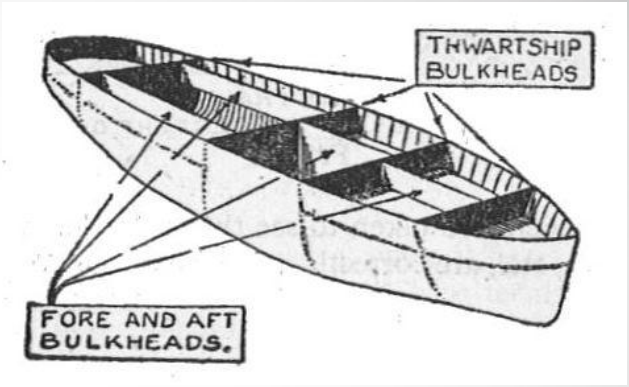
- Applies the retry pattern with a set number of attempts, timeout, and failure threshold.
- Takes an action identifying the process as failed.
- Can involve throwing a custom error or modifying state.

Idempotency pattern

- Design a process where if it is executed repeatedly it produces the same result (not a compounded result).
- Necessary for processes that may be repeated, either due to user action or a connection retry.
- Important for avoiding duplicate database records.
- Consider a configuration-based approach for these processes.
  - A class or function which can be repeatedly called when given the same parameters and will not produce repetitive action.

Bulkhead pattern

- Isolation of a failure to reduce impact on the rest of the system.
- Consider opportunities for redundancy, like a CDN or saving to localStorage until a database connection can be re-established.



Distribution Phase

- Continuous deployment to one or more servers whenever code is saved online.
- Notifying users of change may reduce service tickets.
- Consider running tests within the deployment service.
- Time saved here expedites urgent changes.
- The production branch must always be ready to distribute.

Maintenance Phase

- The longest phase - aided by resilience.
- Observability and notification channels are critical.
- Know about an issue before your users do.
- Be transparent when issues occur for reputation resilience.
- Use a risk register to ensure the right people are notified.

Texas A&M Transportation Institute

Monitoring

Azure notifications and analytics, application logging, and email

Azure

- We use Azure to track application performance and HTTP errors experienced by users
- Billing notifications for Serverless tier SQL Server due to flexible pricing and automated performance scaling
- Twilio Sendgrid for email notifications
- VSCode extension for Azure to quickly examine application logs

Application logging

- For database errors and other user-facing errors.
- Insight into how processes were improved and where improvement is needed.
- Confirm that resilience measures are working as expected.

Email

- Long-running processes email users when complete.
- When they encounter an error an email is sent to notify us.

Takeaways

- Impact determines priority.
- Be aware of design patterns for resiliency.
- Implement monitoring.
- This feature takes time and requires planning.

Know WordPress and want to learn apps?
Join our team!
tx.ag/webgig