**Technical Guidelines**

**API (Protocol)**

1.Restful API : [https://mysite.com/products ความหมาย ไปที่ mysite.com](https://mysite.com/products%20ความหมาย%20ไปที่%20mysite.com)

โดยยิงไปที่ part /products (Back end จะต้องเตรียม URL ให้กับ Front end)

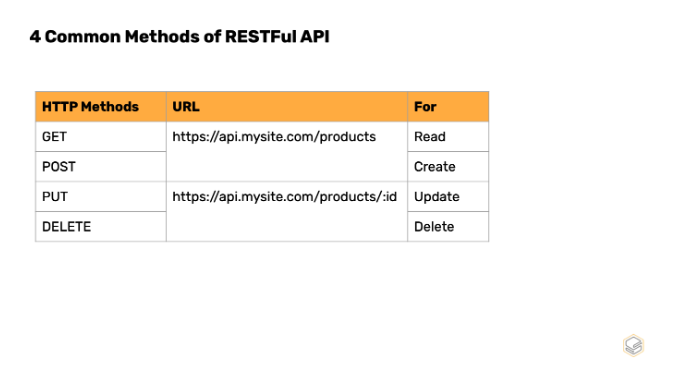
**CRUD**

- **POST** สร้างข้อมูล ยิง API ไปที่ Back end สร้างข้อมูลและรีเทิร์นค่ากลับมาแสดง Front end (Create)

- **GET** ยิง API ไปที่ Back end รีเทิร์นค่ากลับมาแสดงที่ Front end (Read)

- **PUT** Front end ยิง API ไปที่ Back end (Update)

- **DELETE** Front end ยิง API ไปที่ Back end (Delete)



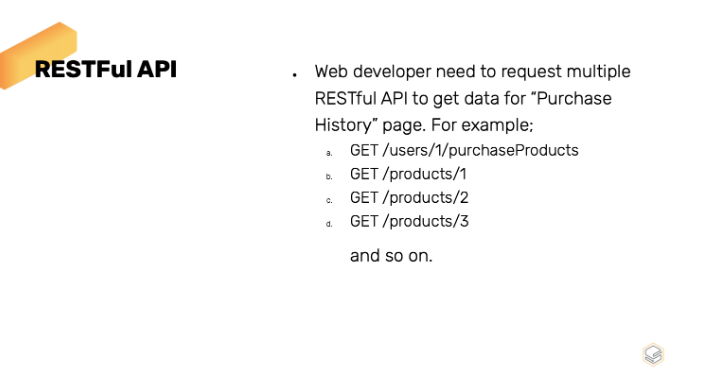
A close-up of a web page

Description automatically generated with low confidence

**RESTFulAPI**

เรียกใช้งาน Http หลาย Part เพื่อแสดงความแตกต่าง

ใช้ Back end ในการกำหนดว่าจะกำหนดข้อมูลใดบ้าง



**GraphQL**

เรียกใช้งาน Http เหมือน RESTFulAPI แต่เรียกใช้งาน Part เดียว

ใช้ Front end ในการกำหนดข้อมูล ต้องการอะไรยิงไปแค่ Part เดียว

A screenshot of a computer

Description automatically generated with low confidence

A picture containing text, screenshot, font, document

Description automatically generated

ข้อดี คือ ดึงข้อมูลไปแสดงแตกต่างกันได้ในหลายๆแพล็ตฟอร์ม

ข้อเสีย คือ **Cache** ยาก เพราะ URL มีแค่ตัวเดียว เก็บค่าแตกต่างกันยาก แต่ข้อดีเยอะกว่าข้อเสีย

Over fetching problem = คือการแสดงข้อมูลที่มากเกินไป เช่น ใน mobile ต้องการดูแค่ชื่อ แต่ดันแสดงข้อมูลทั้งหมด

A picture containing text, screenshot, font, design

Description automatically generated

Multiple round-trips problem = แก้ปัญหาในเรื่องของการยิงหลายๆ URL เป็นการยิงแค่ครั้งเดียว

A picture containing text, screenshot, businesscard, font

Description automatically generated

**Clean Code & Software Design Pattern**

* Good Variable Name - ตั้งชื่อให้อ่านง่าย ให้คนอื่นมาอ่านแล้วเข้าใจหน้าที่ของมัน
* Avoid Magic Number - ประกาศชื่อตัวแปรให้กับค่าคงที่ต่างๆ เพื่อให้รู้ที่มาที่ไป
* Keep It Simple and Stupid - หลีกเลี่ยงการเขียนโค้ดที่ซับซ้อน หรืออ่านยาก
* Don't Repeat Yourself - หลีกเลี่ยงการเขียนโค้ดซ้ำๆเดิมๆที่ทำงานแบบเดียวกัน ควรจะแยกมันเป็นฟังก์ชันที่ทำหน้าที่ชัดเจน
* Single Responsibility - Function หนึ่งๆควรจะมีหน้าที่หลักแค่หนึ่งอย่าง เป็นหนึ่งในหลักของ SOLID Principle

Testing

Unit test เราจะเทสแค่ระบบเดียว หรือเทสแต่ละ Function แต่ละ Class

A picture containing text, screenshot, diagram, font

Description automatically generated

Integration Tset เราจะเทส2ระบบขึ้นไป หรือเทสแต่ละ Function แต่ละ Class โดยมีความสัมพันธ์กับ Function อื่นๆด้วย

A diagram of a test

Description automatically generated with low confidence

End to end Test เป็นการจำลองระบบเสมือนจริง

A picture containing text, screenshot, diagram, font

Description automatically generated

ค่อยๆเทสจากฐานขึ้นมาบนปลายปิระมิด

A picture containing text, screenshot, font, diagram

Description automatically generated

คนส่วนมากจะใช้วิธี Ice Cream Cone Antipattern ในการเทสระบบ ซึ่งเสียค่าใช้จ่ายเยอะรันโค้ดช้า

A picture containing text, screenshot, font, logo

Description automatically generated

**version control**

**Git คือ** Git เป็นเครื่องมือที่ช่วยจัดการเวอร์ชันของโค้ดให้กับเรา โดยจะเก็บประวัติว่าไฟล์แต่ละไฟล์ถูกสร้าง/ลบ/แก้ไขโดยใคร เมื่อไหร่ และอย่างไรเอาไว้ทั้งหมด ทำให้เราสามารถติดตามการเปลี่ยนแปลงของโค้ดได้ตลอด หรือแม้กระทั่งย้อนเวลาโค้ดกลับไปก่อนตอนที่จะพังก็ยังทำได้

****

**GitHub** GitHub คือ website Git (version control repository) ที่อยู่บน internet มีการทำงานแบบเดียวกับ Git เลย แต่สามารถเข้าถึงข้อมูลและจัดการไปผ่าน web โดยไม่ต้องเสียเงิน หรือลงทุกตั้ง server เพื่อติดตั้ง Git เองเลย แต่ code project ทั้งหมดจะถูกแจกจ่ายให้คนอื่นๆสามารถเห็นได้ด้วย

A screenshot of a computer

Description automatically generated with medium confidence

**Git Branches** คือการทำงานแบบแยกส่วน กิ่งก้านออกไป

A diagram of a house

Description automatically generated with low confidence

**Merge** คือการประกอบรวมกัน

A diagram of a house

Description automatically generated with low confidence

**Code Reviews**

**PR =** pull request GitHub

Pull Request เรียกสั้นๆอีกอย่างว่า PR เป็นหนึ่งในช่องทางที่จะสื่อสารกับคนภายในทีมได้ว่า โค้ดจาก Feature branch ที่เราจะเอาขึ้นนั้นมีการเปลี่ยนแปลง อะไรบ้าง และยังเป็นช่องทางที่เอาไว้สำหรับสนทนากันระหว่าง Reviewers (ผู้ตรวจสอบ) และ Author (ผู้สร้าง PR) โดยประเด็นหลักๆที่เราจะใช้ในการสนทนาจะมีประมาณนี้.

* ดูว่า Code มีสิ่งที่ไม่จำเป็นขึ้นไปหรือไม่
* เช็คความเรียบร้อยของงานว่ามีความถูกต้องหรือไม่
* แนะนำการเขียน Code แบบที่ถูกต้อง
* ยิงคำถามถึง Reviewers ในบางครั้งเราก็ไม่มั่นใจว่า Function ที่เราเขียนขึ้นมานั้นดีไหม เราสามารถสร้าง PR ให้เหล่าพี่ช่วยให้คำแนะนำได้
* ตรวจสอบทิศทางการเขียน Code ว่าถูกต้องตามมาตรฐานของทีมหรือไม่ พูดง่ายๆก็คือ เขียน Code เหมือนชาวบ้านหรือไม่
* ช่วยการ Run test ที่อยู่บน CI

A picture containing diagram, text, screenshot, font

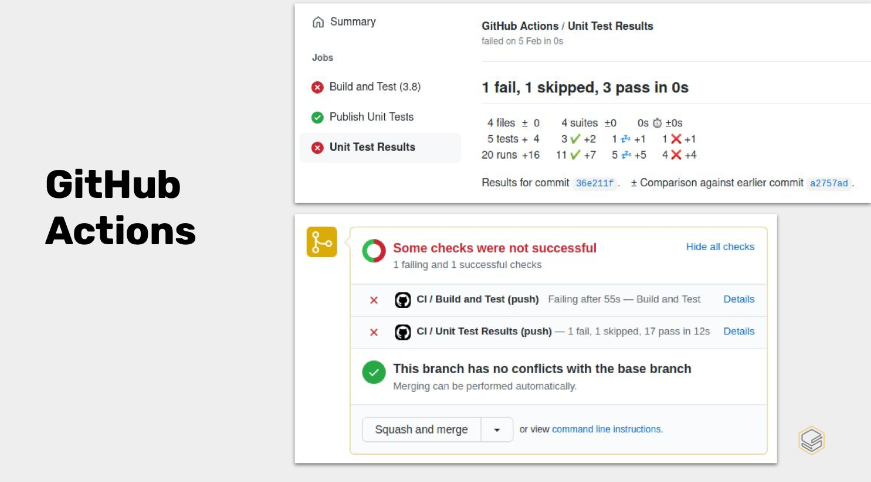
Description automatically generated

**Software Delivery**

**Continuous Integration (CI) คืออะไร มีหน้าที่อย่างไร?**  
  
CI จึงเข้ามามีบทบาทในการช่วย Developer ให้ Merge code ที่แก้ไขส่งกลับไปที่ Pool กลาง เช่น Git Repository ซึ่งอาจจะตั้งค่าให้ทำวันละครั้งก็ได้ ทีนี้เวลาที่ Developer มีการเปลี่ยนแปลงอะไรเกี่ยวกับ App ก็จะมี CI ช่วย Merge code รวมไปถึงการทำ Automate test, Unit test และ Integration test เพื่อให้มั่นใจว่าสิ่งที่ Developer เปลี่ยนแปลงไปจะไม่กระทบกับการทำงานของ Application จะให้ได้ว่ามีการทดสอบหลาย ๆ อย่าง ใช่ครับ ทดสอบทุกอย่างตั้งแต่ Class ไปจนถึง Function ใน Module ที่แตกต่างกันแล้วนำมาประกอบเป็น Application ถ้าการทำ Automate test แล้วเจอปัญหาเรื่อง Conflict ระหว่าง Code เดิมกับของใหม่ CI จะช่วยให้ง่ายต่อการหา และแก้ไข Bug ครับ

**Continuous Delivery (CD) คืออะไร มีหน้าที่อย่างไร?**  
  
หลังจากที่มีการทำ Automation ในเรื่องของการ Build test, Unit test และ Integration test ในฝั่ง CI กันไปแล้ว CD ที่เป็น Continuous Delivery จะทำหน้าที่ส่ง Code ที่มีการตรวจสอบแล้วไปยัง Repository ดังนั้นถ้าต้องการให้ระบบ Continuous Delivery มีประสิทธิภาพมากที่สุด จะต้องทำระบบ CI ให้อยู่ให้ Pipeline เดียวกันกับ CDเป้าหมายของการทำ CD คือการที่พร้อมที่จะนำ Code ไป Deploy ที่ Production ได้ตลอดเวลานั่นเองครับ แต่ Continuous Delivery จะยังเป็น Step Manual อยู่ ไม่ได้เป็น Automation ทั้งหมด

**Continuous Deployment (CD) คืออะไร มีหน้าที่อย่างไร?**  
  
CD อีกตัวนึงที่ต่าง Continuous Delivery นั่นก็คือ Continuous Deployment ขั้นตอนนี้เป็นขั้นตอนสุดท้ายของ CI/CD Pipeline ซึ่งเป็นส่วนที่ขยายมากจาก Continuous Delivery ครับ ซึ่งจะทำหน้าที่จัดการโดยอัตโนมัติในส่วนของการนำ Code จาก Repository ไป Deploy บน ProductionContinuous Deployment จะเหมาะกับการใช้งานที่มีการ Deploy เป็นจำนวนมาก และมีขั้นตอน Test automation ที่ออกแบบมาเป็นอย่างดีโดยในทางปฏิบัติแล้ว Continuous Deployment หมายถึงการที่ Developer สามารถ Dev app ขึ้นมาแล้วไป Go Live บน Cloud ได้โดยใช้เวลาเพียงไม่กี่นาที (อันนี้สมมุติว่าทดสอบด้วยระบบ Automate ผ่านด้วยนะ ) การที่สามารถทำแบบนี้ได้จะช่วยให้ได้รับ Feedback ต่าง ๆ กลับมาไว ไม่ว่าจะเป็นจากฝั่ง User หรือว่า QA ทีนี้เวลาเราเอา CI กับ CD มารวมกัน เป็น CI/CD ก็จะช่วยให้สามารถ Deploy app โดยลดความเสี่ยงและปัญหาที่จะเกิดขึ้นได้ อย่างไรก็ตาม CI/CD จะเหมาะกับการแก้ไขหรือเปลี่ยน Feature เล็ก ๆ น้อย ๆ มากกว่าแก้ไขทั้งหมด แต่การที่จะทำ CI/CD Pipeline ขึ้นมาจะต้องลงแรง และใช้เวลาเนื่องจากต้องวางแผนเรื่อง Automated test ซึ่งมีการเขียนที่หลากหลายรูปแบบตามสถานการณ์ในการทดสอบนั่นเองครับ



A close-up of a computer

Description automatically generated with low confidence