Spesifikasi Tugas Besar 2 IF3110

Milestone 2 - Web Services using SOAP and REST

Deskripsi Persoalan

“Zhisuka menjelaskan mengapa ia suka Dorayaki rasa Kecap - 2021”

Bisnis yang dibuat oleh **Mobita** laris manis akibat digital marketing melalui website yang kalian buat kemarin. Mobita sangat senang karena omset penjualannya melebihi ekspektasinya. Tiba-tiba, **Doremonangis** menelpon Mobita karena mendapati suatu masalah yang besar.

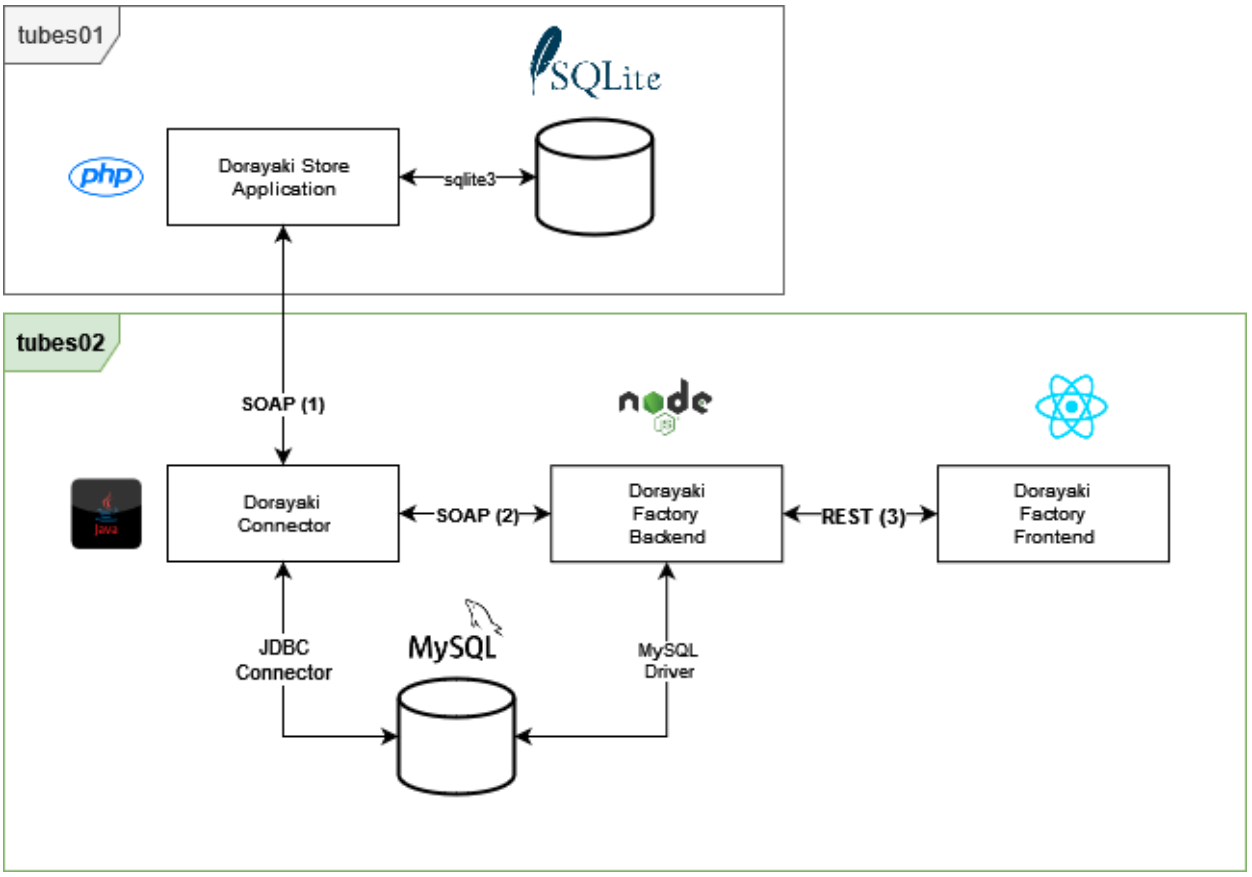
Singkat cerita, toko dorayaki sering kehabisan stok karena *demand* dorayaki yang begitu tinggi. Untuk itu, Mobita berencana untuk membuat pabrik dorayaki dalam skala besar. Dan mengangkat **Zhisuka** untuk menjadi COO karena kebetulan Zhisuka ~~adalah simp Mobita~~ sedang membutuhkan pekerjaan.

Mobita tidak ingin Zhisuka kelelahan mengerjakan operasi secara manual. Oleh karena itu, sebagai teman Mobita yang berkuliah di バンドン工科大学 (ITB) cabang Shinjuku yang telah membantu membuat website yang sangat sukses, Mobita meminta bantuan Anda sekali lagi untuk membuat sistem ini, dengan imbalan berupa voucher Dorayaki rasa ikan asin seumur hidup.

Tujuan Pembelajaran

- Mahasiswa mampu membuat sebuah web service dengan protokol SOAP.
- Mahasiswa mampu membuat sebuah web service yang mengimplementasikan REST.
- Mahasiswa mampu membuat sebuah aplikasi web yang memanggil web service yang mengimplementasikan REST.
- Mahasiswa mampu membuat sebuah aplikasi web yang memanggil web service dengan protokol SOAP.

Spesifikasi Sistem



Gambaran Umum Sistem Dorayaki untuk Milestone 2.

I. Spesifikasi Umum

Sebagai gambaran umum, sistem ini merupakan sebuah sistem pabrik dorayaki yang digunakan untuk menyuplai website toko dorayaki yang sudah kalian buat sebelumnya. Berikut ringkasan hal-hal yang akan kalian kerjakan pada tugas ini:

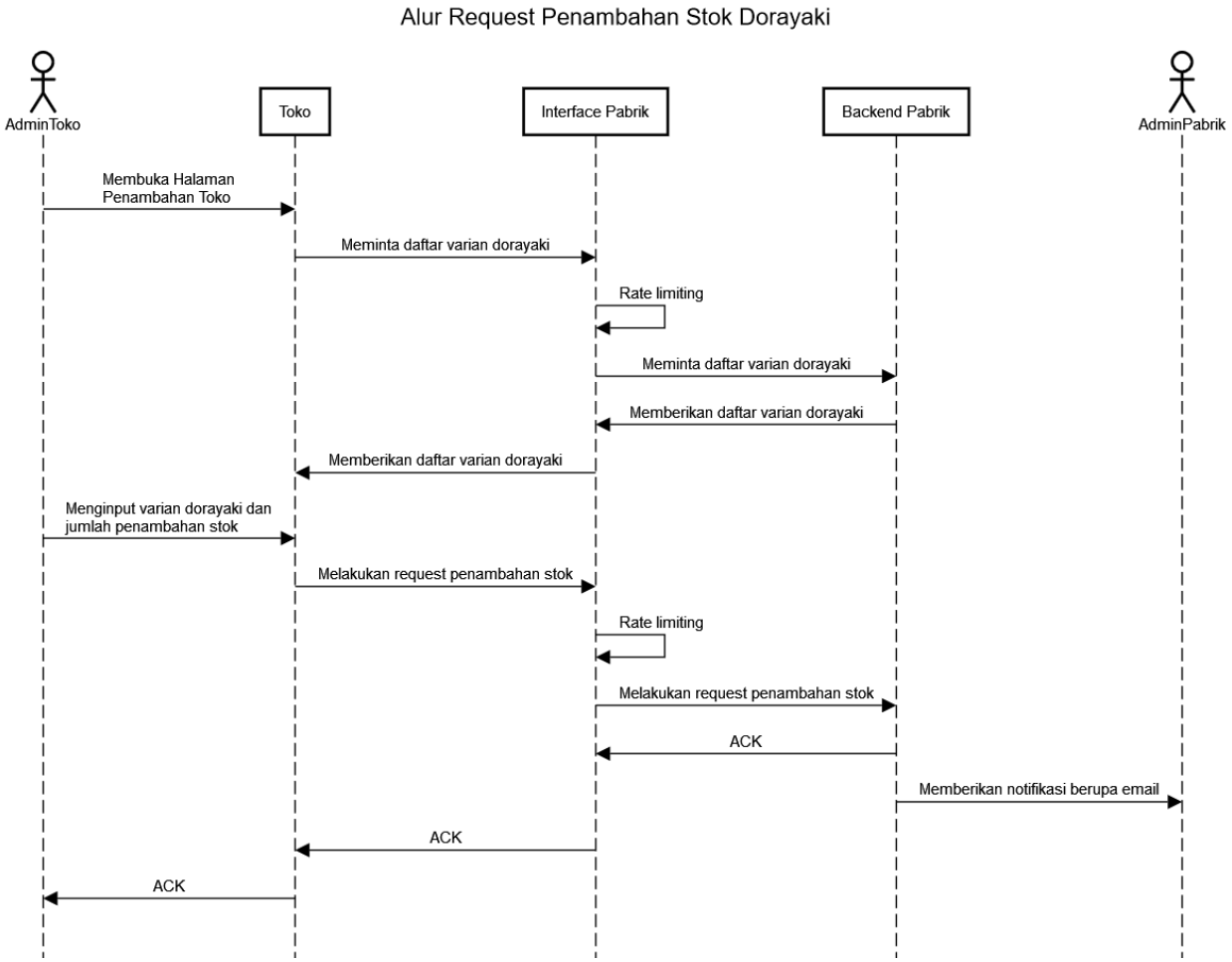
1. Membuat service pabrik dorayaki (frontend dan backend) menggunakan **ReactJS** dengan protokol **REST API** dalam **NodeJS**.
2. Membuat interface pabrik dengan toko menggunakan **JAX-WS** dengan protokol **SOAP** dalam **Java Servlet**.
3. Memperbarui halaman pengubahan stok di website toko agar terhubung dengan service pabrik menggunakan protokol **SOAP** melalui **interface** pabrik.

Diperbolehkan untuk menggunakan library eksternal untuk NodeJS dan React. Usahakan library yang digunakan seminimal mungkin (dalam kata lain, jangan sampai layanan yang dibuat *bloated*). Pada bagian "Bantuan" di spesifikasi tugas, diberikan beberapa opsi library menurut kami menarik untuk dicoba dan dieksplorasi.

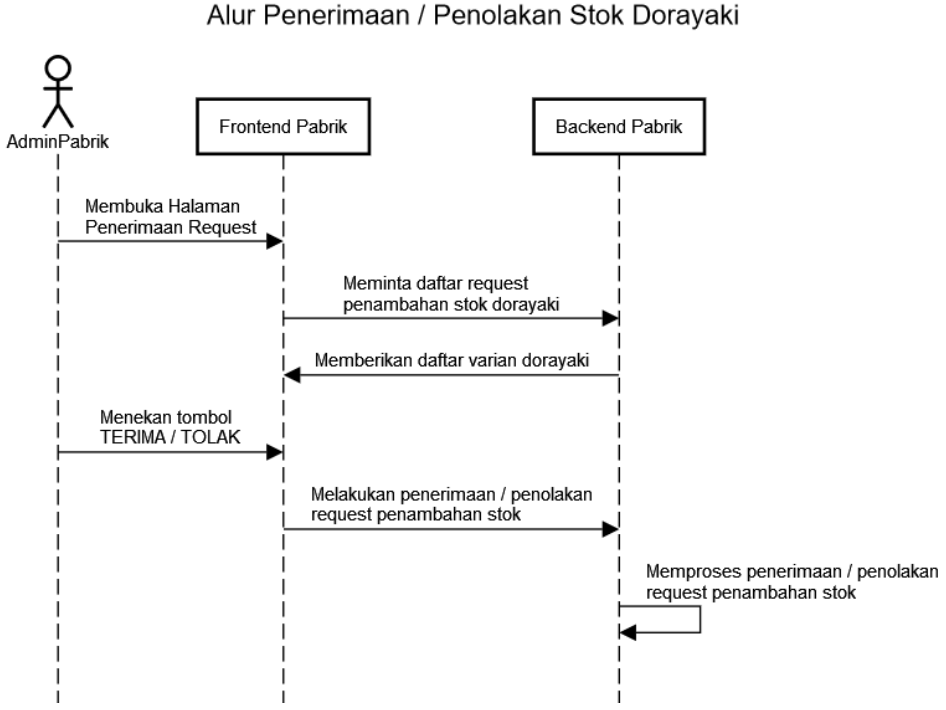
II. User Story

Membangun sistem yang dapat menyuplai website toko dorayaki.

1. Admin toko dapat melakukan request penambahan stok dorayaki ke service pabrik.
2. Admin pabrik dapat melihat daftar request penambahan stok dorayaki.
3. Admin pabrik dapat menerima / menolak request penambahan stok dorayaki.
4. Admin pabrik dapat melakukan pengelolaan bahan baku dorayaki.
5. Admin pabrik dapat melakukan pengelolaan resep dorayaki.
6. Admin pabrik menerima notifikasi melalui email ketika mendapat request penambahan stok dorayaki baru



Alur Request Penambahan Stok Dorayaki



Alur Penerimaan Request Stok Dorayaki

III. Spesifikasi Fitur Pabrik; Backend Service Pabrik

Berikut fitur-fitur yang tersedia di dalam service pabrik:

1. **Database Pabrik**

   Basis data menggunakan **MySQL**. Skema basis data pabrik **dibebaskan**, asalkan dapat memenuhi setiap requirement yang ada. Secara garis besar, akan terdapat tabel **resep**, **bahan baku**, **request** dari toko, dan **log request** dari toko.

   Tabel log request **dibedakan** dari tabel request untuk memenuhi fitur pembatasan request yang dijelaskan pada bagian IV.3.

   Beberapa tips untuk meningkatkan performa database:

   a. Usahakan untuk **meminimalkan** jumlah pemanggilan basis data dan juga melakukan optimasi query. Umumnya, menjalankan query pada basis data merupakan operasi yang mahal,
   b. Lakukan **indexing** pada kolom tabel yang sering dilakukan filtering (WHERE clause). Contoh query "SELECT id, name FROM user WHERE name LIKE 'J%'", maka sangat disarankan untuk meng-index kolom name,
   c. Tips lain dapat dicari dengan kata kunci "database optimization"

2. **Autentikasi Pengguna (Admin Pabrik Dorayaki)**

   Tidak seperti toko yang memiliki 2 tipe pengguna, kali ini hanya ada 1 tipe pengguna yang dapat mengakses sistem pabrik, yaitu admin. Autentikasi dibangun dengan standar JWT atau OAuth 2.0 (diperbolehkan menggunakan library external).

3. **Pengelolaan Request Penambahan Stok**

   Sistem pabrik dapat menerima permintaan penambahan stok dari sistem eksternal (toko). Admin pabrik dapat:

   - **Melihat daftar** request penambahan stok dorayaki.
   - **Menerima (accept) / menolak (decline)** request penambahan stok dorayaki.

   Ketika admin menerima (accept) request, **stok bahan baku otomatis berkurang**, namun stok dorayaki **di toko tidak langsung bertambah** (di luar kepentingan pabrik). Pabrik hanya perlu mengupdate status request dan toko akan secara berkala memantau status request yang pernah diajukannya.

4. **Manajemen Resep**

   Untuk membuat sebuah varian dorayaki, sistem pabrik memiliki database resep. Resep dorayaki hanya menyimpan daftar bahan baku dan jumlah bahan baku yang diperlukan. Tidak perlu menyimpan cara mengolah dorayaki, tahapan pembuatan dorayaki, dsb.

Admin pabrik:

- **Dapat membuat** resep varian dorayaki baru
- **Dapat melihat** daftar seluruh resep dan isinya (boleh diletakkan di 1 halaman yang sama maupun dipisah url berbeda)
- **TIDAK DAPAT menghapus** resep, karena akan menimbulkan konflik ketika toko sedang melakukan request dorayaki tersebut.
- **TIDAK DAPAT mengubah** isi resep dengan alasan yang sama.

5. **Manajemen Bahan Baku**

   Sistem pabrik memiliki database bahan baku yang dimiliki. Pabrik mencatat **nama** dan **stok** bahan baku pada setiap waktu. **Tidak** seperti toko, riwayat perubahan bahan baku tidak perlu disimpan.

   Admin pabrik:

   - **Dapat membuat** entri bahan baku baru
   - **Dapat melihat** daftar bahan baku dan stoknya
   - **TIDAK DAPAT menghapus** bahan baku yang sudah ada, namun
   - **Dapat mengupdate** bahan baku (atribut yang dapat diubah dibebaskan, **minimal** dapat mengubah **stok**nya)

6. **Notifikasi Email**

   Ketika terdapat request penambahan stok baru dari toko, dikirimkan email ke akun admin tertentu (boleh seluruh akun, boleh salah satu akun saja, dibebaskan). Pengiriman email dapat dilakukan dengan pustaka *nodemailer* maupun pustaka lainnya.

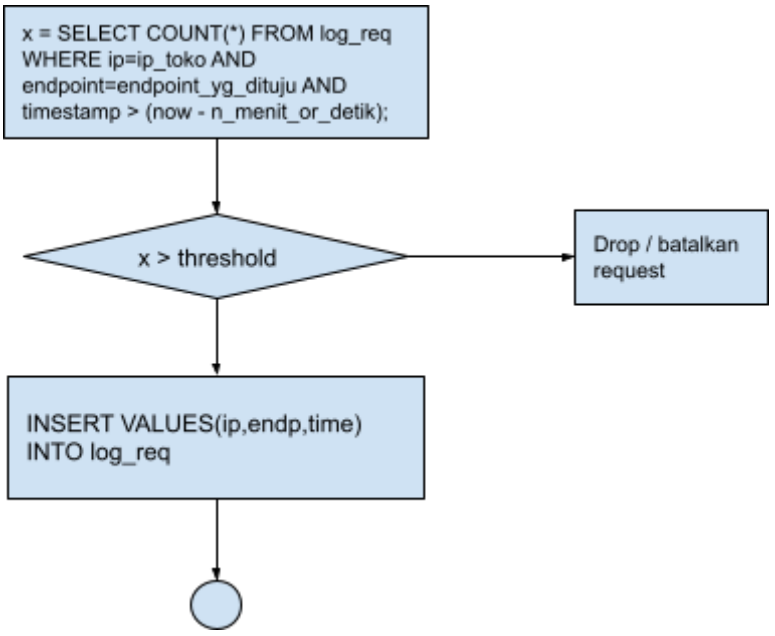
IV. Spesifikasi Interface Pabrik

Agar toko dapat berinteraksi dengan pabrik, sistem pabrik memiliki service khusus untuk meng-*expose* beberapa fungsi dari pabrik, yaitu:

a. Melakukan request pengiriman dorayaki (penambahan stok)
b. Membaca status request pengiriman yang dilakukan toko tersebut
c. Membaca varian dorayaki yang disediakan pabrik

Service dibangun dengan menggunakan **JAX-WS** dengan protokol **SOAP** pada **Java Servlet**. Interface harus mampu:

1. **Menyediakan WSDL** bagi toko **yang terdaftar** untuk setiap endpoint yang dimiliki.
2. **Menerima SOAP Request** dari toko lalu "meneruskannya" ke backend service dengan cara memanggil endpoint backend service (dan menyusun / meng-*compose* response, jika dibutuhkan).
3. Melakukan **pembatasan request** / *rate-limiter* sederhana (perhatikan gambar di bawah). Pembatasan request **harus** dilakukan dengan cara:
    i. Mengecek apakah jumlah request sudah melebihi *threshold*, misal 10 request per menit. Pengecekan dilakukan dengan melakukan **count** row tabel log request.
    ii. Apabila tidak melebihi threshold, lakukan logging setiap request yang dilakukan oleh toko sebelum memproses request. Data yang disimpan adalah **ip**, **endpoint** yang dituju, **timestamp** saat itu.
    iii. Apabila sudah melebihi threshold, *drop* / batalkan request.



Flowchart Mekanisme Rate-limiter

V. Spesifikasi Tiap Halaman; Frontend Service dan Service Toko

Implementasi dan desain pada service pabrik secara umum lebih bebas daripada tugas pertama, asal dapat memenuhi kriteria di bawah ini:

1. **Perubahan Halaman Edit Stok di Toko**

   Service toko tidak lagi secara mandiri melakukan pengelolaan stok. Admin **tidak dapat** melakukan **perubahan stok**, baik menambah maupun mengurangi stok. Namun, admin dapat menambah stok dengan melakukan **SOAP request penambahan stok** ke **interface pabrik**. Dorayaki yang tidak disediakan oleh pabrik, tidak bisa ditambah lagi stoknya.

2. **Perubahan Halaman Create Dorayaki di Toko**

   Dorayaki yang dapat ditambahkan **terbatas** pada dorayaki yang disediakan oleh pabrik. Toko perlu mendapat informasi varian yang disediakan pabrik melalui SOAP request. Kemudian, nama dorayaki diambil dari nama yang disediakan oleh pabrik, sedangkan informasi lain seperti deskripsi, harga, dan gambar boleh diisi sendiri oleh admin toko.

3. **Halaman Login dan Register di Pabrik (1 halaman masing-masing)**

   Seperti halaman login / register pada umumnya. Desain dibebaskan.

4. **Halaman Resep di Pabrik (1 halaman atau lebih)**

   Halaman ini dapat **menampilkan daftar resep**, **membuat resep baru**, dan **menampilkan detail resep**. Diperbolehkan untuk membuat 1 halaman saja (digabung semua untuk ketiga fitur di atas) ataupun dipisah menjadi 2 atau 3 jenis halaman berbeda.

5. **Halaman Bahan Baku di Pabrik (1 halaman atau lebih)**

   Seperti halaman resep, namun di halaman ini bahan baku dapat diubah atributnya, minimal mengubah stoknya.

6. **Halaman Daftar Request di Pabrik (1 halaman)**

   Pada halaman ini data daftar request ditampilkan dalam bentuk tabel yang terdapat tombol / mekanisme untuk melakukan perubahan status tiap requestnya. Aksi yang dapat dilakukan berupa **menerima (accept) / menolak (decline) request**. Tabel memiliki fungsi search dan sort by each specified column (minimal kolom status dan datetime dilakukannya request). Ingat, kalian boleh menggunakan external library untuk membuat tabel.

Bonus

Catatan: Kerjakan dahulu spesifikasi wajib sebelum mengerjakan bonus.

1. **Docker**

   Membuat **Dockerfile** dari aplikasi kalian, serta membuat file **docker-compose.yml** dari aplikasi kalian yang berisi aplikasi kalian serta database yang digunakan. Tidak perlu membuat dockerfile yang kompleks, asalkan bisa dijalankan. Tujuan bonus ini adalah agar kalian dapat mendapatkan pengalaman *hands-on* menggunakan docker.

   Sebagai catatan, untuk men-*dockerize* React usahakan untuk melakukan serving hasil static build aplikasi kalian.

2. **Caching Query / Response**

   Melakukan query ke database merupakan operasi yang mahal. Caching akan membantu meningkatkan performa aplikasi kalian secara signifikan pada traffic yang tinggi. Caching dapat dilakukan pada level query maupun pada level response sebuah request. Service / stack yang umum digunakan untuk caching adalah redis.

Bantuan

Tautan Tutorial yang __*semoga*__ Membantu

Untuk membantu anda dalam mengerjakan tugas ini, ada beberapa tautan yang bisa anda buka untuk menyelesaikan tugas ini.

- https://www.codejava.net/java-ee/web-services/how-to-code-and-deploy-java-xml-web-services-jax-ws-on-tomcat
- https://examples.javacodegeeks.com/enterprise-java/jws/jax-ws-tutorial-beginners
- https://www.baeldung.com/jax-ws

(iya, setup java servletnya memang susah, semangat ya)

(iya, tutorialnya banyak, caranya beda-beda semua)

Daftar Library

Berikut ada beberapa library yang disarankan oleh asisten. Kalian tidak harus mengikuti apa yang disarankan di sini, tapi bisa menjadikan ini referensi.

Untuk Java, diwajibkan menggunakan Jax-WS, sehingga kami tidak akan menyarankan library yang lain.

Untuk NodeJS, ada beberapa library yang disarankan:

1. Express (https://expressjs.com/), web framework NodeJS yang paling terkenal.
2. Fastify (https://www.fastify.io/), web framework yang low-overhead dan secara benchmark mengalahkan Express.
3. Hapi (https://hapi.dev/), web framework dengan tujuan untuk menjaga keamanan

Untuk React, diperbolehkan untuk menggunakan library styling apapun (seperti TailwindCSS, WindiCSS, Bootstrap, Bulma, dll.). Untuk development server dan hal terkait ada beberapa opsi yang disarankan:

1. create-react-app/CRA (https://create-react-app.dev/), cara standar yang disarankan dokumentasi React untuk membuat aplikasi React yang menggunakan Webpack.
2. Vite (https://vitejs.dev/), menyediakan development server yang menggunakan Rollup dan lebih cepat dibandingkan CRA.
3. Melakukan setup lewat Webpack sendiri.
4. Melakukan serving React melalui server framework seperti Express.

Typescript diperbolehkan dan sangat disarankan. (https://www.typescriptlang.org/)

Responsi

Akan diadakan responsi per kelompok bersama asisten menjelang pengumpulan tugas (tanggalnya menyusul, kemungkinan di hari Sabtu seminggu sebelum deadline).

Responsi bersifat **opsional**. Kelompok yang ingin mendaftarkan diri dapat mengisi pada sheet yang ada di MS Teams.

Perlu diperhatikan bahwa ketersediaan asisten **terbatas**. Siapkan terlebih dahulu hal-hal yang ingin ditanyakan / dikonsultasikan dan lakukan setup project di komputer anda **sebelum** sesi responsi dimulai.

Daftar Pertanyaan

Jika masih ada pertanyaan mengenai spesifikasi tugas, dapat dilakukan melalui sheet yang ada di MS Teams. Pastikan tidak ada pertanyaan yang berulang.

Lain-Lain

Deliverables

Berikut adalah hal yang harus diperhatikan untuk pengumpulan tugas ini:

1. Buatlah grup pada Gitlab dengan format "IF3110-2021-02-YY", dengan YY adalah nomor kelompok.
2. Tambahkan anggota tim pada grup anda.
3. Buatlah **empat** repository private berikut pada grup tersebut:
    a. Dorayaki-Store (PHP + HTML-CSS-JS, gunakan aplikasi Dorayaki Store salah satu anggota anda sekarang)
    b. Dorayaki-Supplier (SOAP, java servlet)
    c. Dorayaki-Factory-Server (REST, node)
    d. Dorayaki-Factory-Client (NodeJS)
4. Silakan commit pada repository anda (hasil fork). Lakukan beberapa commit dengan pesan yang bermakna, contoh: “add register form”, “fix logout bug”, jangan seperti “final”, “benerin dikit”, “fix bug”. Disarankan untuk tidak melakukan commit dengan perubahan yang besar karena akan mempengaruhi penilaian (contoh: hanya melakukan satu *commit* kemudian dikumpulkan). Sebaiknya *commit* dilakukan setiap ada penambahan fitur. *Commit* dari setiap anggota tim akan mempengaruhi penilaian. Jadi, setiap anggota tim harus melakukan commit yang berpengaruh terhadap proses pembuatan aplikasi. Sebagai panduan bisa mengikuti semantic commit.
5. Perbaruilah file README di repository Dorayaki-Store dengan:
    a. Perubahan skema basis data, jika ada
    b. dan perubahan lainnya yang ingin anda sebutkan
6. Tambah file README di repositori Dorayaki-Supplier (SOAP) dan Dorayaki-Factory-Server (REST), minimal berisi:
    a. Deskripsi singkat web service
    b. Skema basis data yang digunakan
    c. (opsional) Endpoint, payload, dan response API
    d. Penjelasan mengenai pembagian tugas masing-masing anggota (lihat formatnya pada bagian pembagian tugas).
7. Tambah file README di repositori Dorayaki-Factory-Client (NodeJS), minimal berisi:
    a. Deskripsi singkat aplikasi
    b. Screenshot tampilan aplikasi (tidak perlu semua kasus, minimal 1 per halaman), dan
    c. Penjelasan mengenai pembagian tugas masing-masing anggota (lihat formatnya pada bagian pembagian tugas).

Pengumpulan Tugas

Deadline tugas adalah pada hari **Jumat, 26 November 2021 pukul 15.00 WIB**. Waktu pengumpulan tugas yang dilihat adalah **waktu push ke server Gitlab** terakhir. Dipilih deadline jam 3 sore karena jika terjadi error pada server Gitlab Informatika, masih ada staff yang dapat menangani hal tersebut.

Pembagian Kelompok

Silakan isi daftar kelompok kalian pada sheet yang terdapat di teams.

Kelompok maksimal terdiri dari 3 orang, diperbolehkan lintas kelas, diperbolehkan berbeda dengan tugas 1. Gunakan salah satu sistem toko anggota kelompok kalian.

Pembagian Tugas

Setiap anggota kelompok **diwajibkan** untuk mengerjakan bagian REST, SOAP, dan NodeJS, dengan harapan kalian bisa mencoba mempelajari semua bagian dan cara kerjanya.

REST

Backend Pabrik Fungsi X : 13519xxx, 13519xxx

Backend Pabrik Fungsi Y : 13519xxx

(Lanjutkan ...)

SOAP

Login : 13519xxx, 13519xxx

Register : 13519xxx

(Lanjutkan...)

NodeJS

Frontend Pabrik Fungsi X : 13519xxx, 13519xxx

Frontend Pabrik Fungsi Y : 13519xxx

(Lanjutkan...)

Lain-lain

Files penting diletakkan di teams untuk kepentingan pengarsipan dokumen / artefak tugas.