Prueba para puesto Tech Lead.

Crack The Code.
Roberto Guzmán Sánchez
Ing. Software.

Task: Migración monolito a microservicios.

Objetivo:

Preparar el ambiente arquitectónico y de software en el backend y el frontend para absorber la demanda de una nueva línea de negocio que se centrará en la captación de B2B de manera asíncrona a la demanda actual del B2C.

Descripción.

Los usuarios B2B son empresas o instituciones que solicitarán de manera masiva usuarios que puedan inscribirse en grupos completos para los diferentes cursos en complejidades variadas, de manera que se esperan desde grupos principiantes, medios y avanzados, identificados principalmente por la edad y año en curso escolar.

La primicia es que sin afectar a los usuarios que llegan de manera individual conocidos como B2C puedan integrarse de forma concurrente y asíncrona.

Restricción.

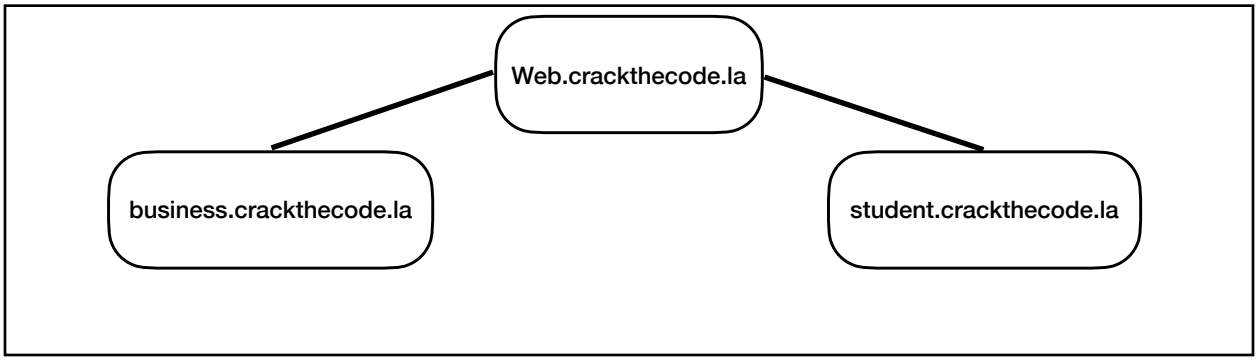
El usuario que fue inscrito por B2B en caso de que la empresa quiera finalizar el contrato, el avance, los proyectos y cursos realizados por cada usuario se perderán con dicho contrato, de manera que los usuarios B2B y los usuarios B2C estarán separados unos de otros.

Desglose.

Actualización en Front.

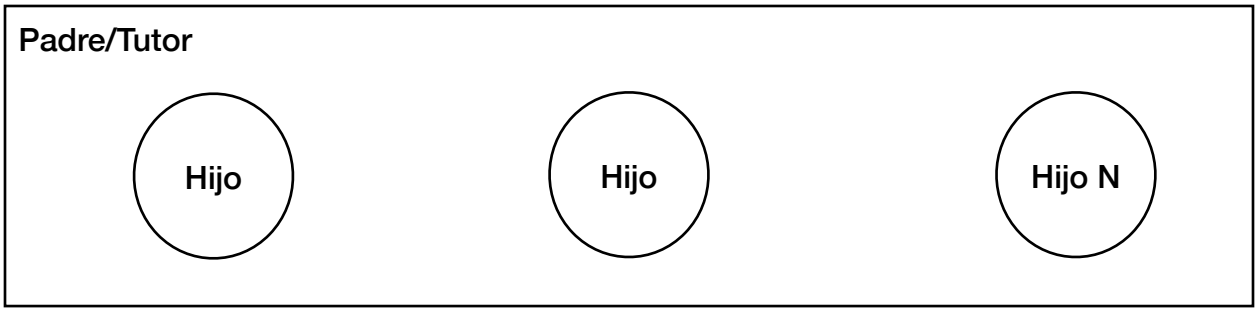
Actualización del sitio web utilizando NextJs para mejorar el SEO y mantener el estilo de programación que se tiene con ReactJs, al mejorar el SEO se reduce el presupuesto para Google Ads y buscadores, aumentando el tráfico de manera orgánica.

Onboarding.



Estructura B2C

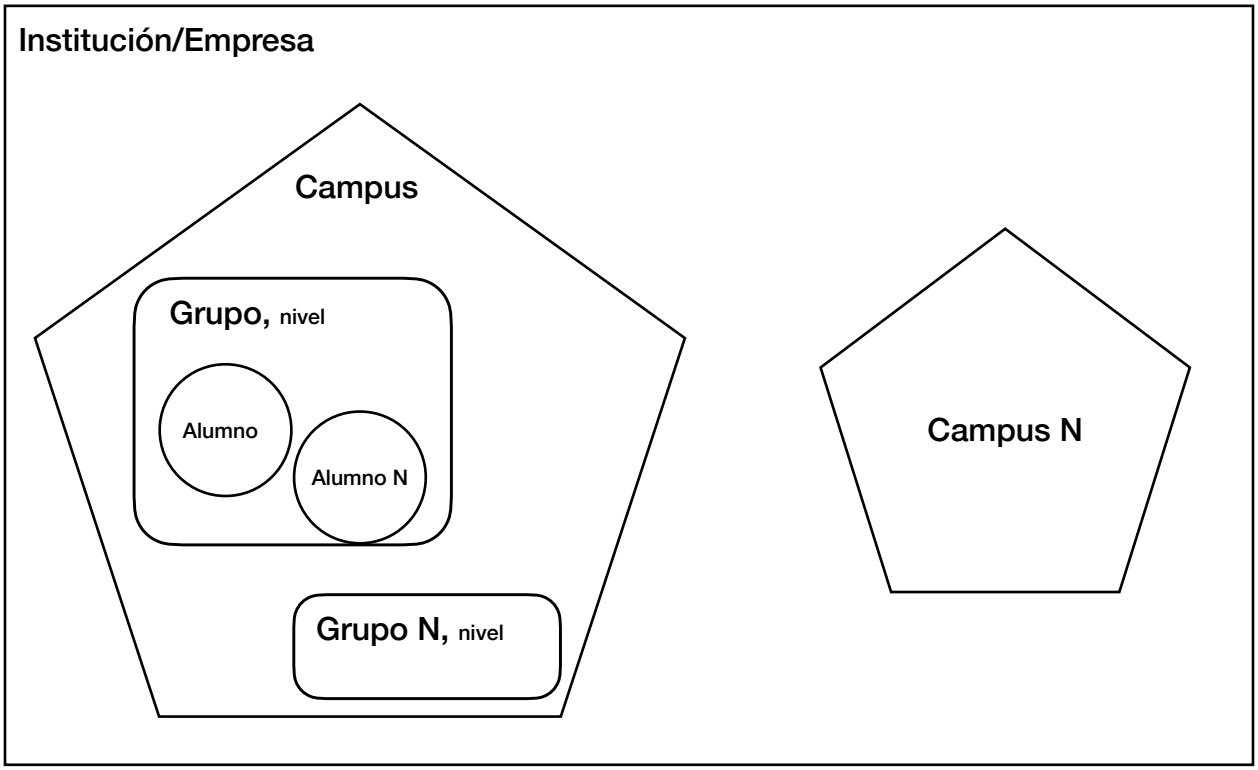
- Padre o Tutor
  - Hijos



Con esta estructura se pretende que el tutor tenga acceso a reportes de performance, pagos y posibilidad de inscripción/baja de los estudiantes inscritos a su cuenta.

Estructura B2B

- Institución o empresa.
  - Campus.
    - Grupos.
      - Estudiantes.



Utilizando la información de esta manera, la empresa cuenta con roles y permisos que le faciliten la gestión e información sobre el avance, facturación y actividad de cada estudiante visto en diferentes niveles para la abstracción de datos.

Microservicios de mayor prioridad.

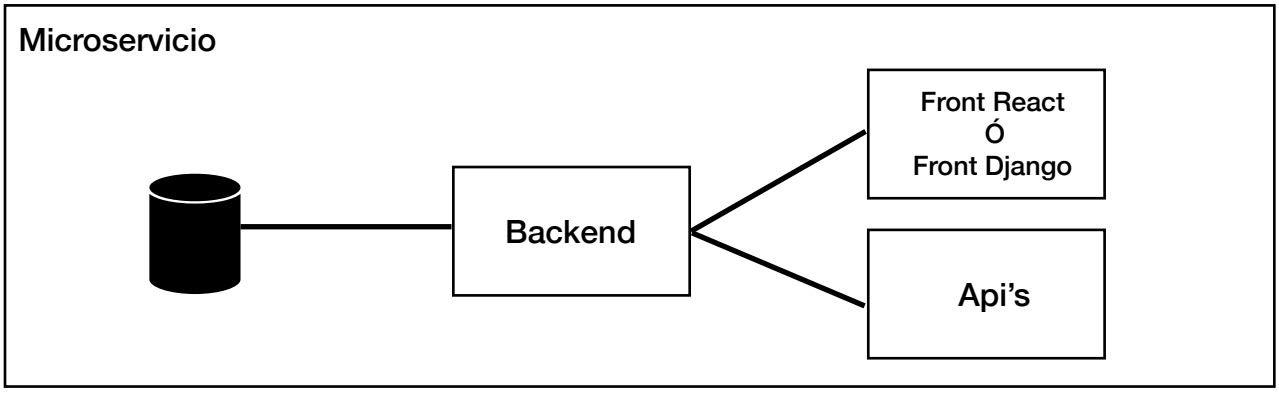
Para el desarrollo de estos microservicios existen dos opciones, dependerá de el trabajo que ya se tenga.

Opción 1 - En caso de que no exista un frontend dedicado para la administración siendo un usuario Crack The Code, entonces se sugiere utilizar Django, esto porque se utilizaría el front que ya ofrece Django para realizar tareas de soporte nivel 1, añadiendo herramientas como retool para la generación de gráficos y reportes.

En este caso para la documentación de las api se puede utilizar Swagger.

Opción 2 - Si el frontend de administración para Crack The Code ya existe entonces se puede utilizar Flask, reduciendo el tamaño y aumentando la flexibilidad del microservicio.

En este caso para la documentación de las api se puede utilizar Flask-rest-api.



1.- Microservicio de cursos.
Al recibir una carga mayor a la habitual en el consumo de los cursos y por ser un servicio streaming, es necesario que este microservicio cuente con una mayor cantidad de recursos en comparación de los demás, de forma que se evite la latencia y sobresaturación en todo el sistema, así que se sugiere separar esta operación del monolito.

2.- Microservicio de Usuarios B2B
Es necesario realizar un microservicio para estos usuarios ya que estarán desde un principio separados de los usuarios actuales obtenidos por el canal B2C, de igual manera, estos usuarios se comportan diferente desde un inicio porque se encuentran anclados a un campus de la institución que tiene el contrato activo, los usuarios creados podrían limitarse con un mínimo y máximo de registros, de manera que siempre exista un super admin que toma un rol similar al del tutor en el caso de B2C, como se describe a continuación:

La institución tiene un *super admin* que permitirá crear N número de campus.
El campus tiene un *super admin Campus* que permitirá crear N número de grupos.
El grupo tiene su *super admin Grupo* que permite crear a N número de estudiantes.

3.- Microservicio de estrutura B2B
Para que la institución pueda gestionar su facturación, es necesario que tenga información clara sobre los estudiantes inscritos, de forma que estos pertenezcan de manera directa a un grupo dentro de un campus, haciendo que dicha empresa pueda abrir N número de campus y

N número de grupos, limitando si es necesario el número de alumnos con un tope máximo y mínimo.

De esta manera los reportes se podrían entregar en diferentes niveles, permitiendo hacerle ver a la empresa el performance por campus inicialmente aunque de manera más detallada conocer la información por grupo y/o por alumno si así se requiere.

- Monolito.
Realizando esta separación en el monolito estaría quedando la operación actual, quitando la carga de trabajo que se tiene por el consumo de los cursos.

OpenAPI

```yaml
openapi: 3.0.3
info:
  title: Students
  version: 1.0.0
  description: Get student list in Crack the code
paths:
  /ctc/studens/api/studens/:
    get:
      operationId: studens_list
      parameters:
      - name: page
        required: false
        in: query
        description: student paging
        schema:
          type: integer
      tags:
      - ctc
      security:
      - cookieAuth: []
      - basicAuth: []
      - ApiKeyAuth: []
      responses:
        '200':
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/PaginatedStudentList'
          description: 'Get list of students in Crack the code'
    post:
      operationId: studens_create
      tags:
      - ctc
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Student'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Student'
          multipart/form-data:
            schema:
              $ref: '#/components/schemas/Student'
        required: true
      security:
      - cookieAuth: []
      - basicAuth: []
      - ApiKeyAuth: []
      responses:
        '201':
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Student'
          description: 'Insert new Crack The Code Studen'
  /ctc/studens/api/studens/{id}/:
    get:
    # ...
```