Interrogación IIC2513 2021-2

> **Fecha de Entrega**: jueves 25 de noviembre de 2021 a las 22:00 horas.

Contexto

El pasado domingo 21 de noviembre se realizaron las elecciones presidenciales, parlamentarias y de CORE en el país. Este proceso electoral fue particularmente importante dados los recientes acontecimientos ocurridos en el país, tales como el estallido social de 2019, la pandemia del COVID-19 y el proceso constituyente que comenzó en julio de 2021. El clima político ha estado bastante agitado y las elecciones presidenciales, en particular, han tomado el protagonismo de la agenda nacional en los últimos meses.

Como una forma de poder establecer diferencias entre los diferentes candidatos y sus propuestas, el equipo docente del curso IIC2513 ha construido una pequeña plataforma web que permite visualizar datos concretos sobre cada uno de los presidenciables. La base de esta plataforma ya ha sido implementada, sin embargo, el equipo docente ha estado bastante ocupado corrigiendo las últimas entregas del proyecto del curso, por lo que no lograron finalizarla previo a las elecciones.

Para no dejar la plataforma abandonada, se le ha encargado a usted seguir con el proyecto y producir una primera versión funcional. En términos técnicos, consta de una aplicación web construida con el framework Koa, utilizando la configuración del template del curso. Se divide en 3 partes: una landing page, una API y una vista que consume la API mediante JavaScript.

El proyecto existente

Como la plataforma web ya se encuentra parcialmente implementada, para el desarrollo usted debe clonar y modificar el proyecto cuyo link puede obtener desde acá. En este repositorio encontrará instrucciones para ejecutar la aplicación.

El modelo de datos consta solamente de dos modelos: `candidate` y `proposal`. Puede revisar en el código cómo están asociados.

Nota: usted **NO DEBE** modificar la implementación de estos modelos

Las siguientes secciones describen lo que usted debe implementar específicamente, dividido en 3 partes.

Parte I: Preguntas teóricas [1.0 pt]

Esta primera parte es la única que no tiene relación con el contexto de la aplicación. A continuación encontrará una serie de preguntas teóricas. Usted debe seleccionar y responder 5 preguntas de las 10 disponibles.

1. ¿Cuál es la diferencia entre los métodos HTTP `PUT` y `PATCH`?

   El método HTTP `PUT` se utiliza para actualizar un recurso por completo **[0.1 pts]**, mientras que el método HTTP `PATCH` se utiliza para actualizar un recurso de forma parcial **[0.1 pts]**. Otra diferencia es que si se hace un request con método `PUT` y el recurso no existe, este se crea, a diferencia de `PATCH` que debería retornar 404.

2. Explique qué significa el status code 422 del protocolo HTTP.

   El status code `422` tiene por nombre "Unprocessable Entity" **[0.1 pts]**, y se refiere a un error de cliente en que, a pesar de que el request tiene la sintaxis correcta, el servidor no es capaz de procesarlo debido a un problema con la entidad asociada **[0.1 pts]** (comúnmente validación).

3. Un documento HTML escrito con etiquetas semánticas (header, article, section, etc) es correcto a diferencia de uno que no incluye este tipo de etiquetas. Comente esta afirmación presentando argumentos.

   La afirmación no es correcta, debido a que un documento HTML escrito **sin** etiquetas semánticas perfectamente puede estar correcto en cuanto a sintaxis **[0.1 pts]**. Lo que las etiquetas semánticas proveen es mayor información respecto al propósito de cada sección del documento, lo que puede ser muy útil para mejorar accesibilidad y también para motores de búsqueda **[0.1 pts]**. Sin embargo, el documento es funcional si no se incluyen.

4. En relación a CSS, mencione 2 diferencias entre las técnicas de disposición de elementos Flexbox y Grid.

   Flexbox permite disponer elementos en un solo eje (horizontal o vertical) mientras que Grid permite hacerlo en dos ejes (especificando filas y columnas) **[0.1 pts]**. Por otro lado, si bien ambos permiten armar grillas, con Grid se tiene más flexibilidad que con Flexbox debido a que fue diseñado para eso, y esto finalmente implica menos código **[0.1 pts]**.

5. Considere el siguiente trozo de código JavaScript.

   ```
   const person1 = {
     name: 'Foo',
     address: {
       street: 'Vicuña Mackenna',
       city: 'Santiago',
     }
   };

   const person2 = {
     name: 'Bar',
     address: person1.address,
   }
   person2.address.city = 'Berlín';

   console.log(person1.address);
   ```

   ¿Por qué vemos que se imprime una dirección de `person1` algo diferente a lo definido originalmente?

   Porque la key `address` tanto de `person1` como de `person2` está apuntando o referenciando al mismo objeto JavaScript **[0.1 pts]**. Por lo tanto, al modificar la propiedad `city` del objeto accediéndolo a través de `person2.address`, se está modificando el mismo objeto que es referenciado por `person1.address` **[0.1 pts]**.

6. Considerando que ambas funcionalidades de JavaScript nos permiten ejecutar código asíncrono, ¿cuál es la diferencia entre callbacks y promesas?

   Los callbacks son funciones que le pasamos a alguna funcionalidad asíncrona, para que se ejecuten usualmente una vez que esta termine **[0.1 pts]**, mientras que las promesas son objetos JavaScript que representan un valor que se obtendrá en el futuro producto de una operación asíncrona **[0.1 pts]** Además, las promesas permiten encadenar una serie de respuestas a operaciones asíncronas, permitiendo manejarlas de forma más limpia.

7. ¿A qué nos referimos cuando decimos que Node.js permite realizar operaciones I/O no-bloqueantes?

   Todas las operaciones I/O en Node.js son asíncronas **[0.1 pts]**, lo que significa que el proceso no se quedará esperando a que una operación de este tipo termine antes de continuar su ejecución, por lo tanto, no bloquea otras operaciones (como recibir nuevos requests HTTP) **[0.1 pts]**. Esto es posible gracias al event loop, que permite ejecutar código asíncrono en un solo thread.

8. ¿Por qué una herramienta como ESLint puede ser útil para un proyecto de software hecho en JavaScript? Mencione al menos 2 razones.

Por un lado porque permite estandarizar el estilo del código, facilitando que distintos desarrolladores puedan colaborar en un mismo proyecto **[0.1 pts]**. Por otro lado, puede ayudar a identificar código eventualmente duplicado y también problemas de sintaxis **[0.1 pts]**.

9. En relación a la arquitectura REST definida por Roy Fielding, ¿a qué se refiere el principio de "mensajes auto-descriptivos" dentro del principio general de "interfaz uniforme"?

   Se refiere a que cada mensaje que se intercambia entre cliente y servidor incluye la información suficiente para ser procesado por el receptor **[0.1 pts]**. En la práctica esto se puede ver en que incluye el formato o "media type" del mensaje (HTML, JSON, XML, etc) **[0.1 pts]**. Puede incluir también información sobre cómo ser "cacheado".

10. ¿Cuál es la importancia del componente "signature" dentro del formato de un JSON Web Token (JWT)?

    El signature de un JWT es fundamental por 2 razones: para asegurar que el mensaje del payload no ha sido modificado por terceros **[0.1 pts]**, y para verificar que el JWT fue efectivamente generado por la entidad que se espera que lo haya hecho **[0.1 pts]**.

Implementación

El detalle de las preguntas lo encontrará también ingresando a la ruta `/preguntas-teoricas` de la aplicación web. Esta ruta está implementada como una vista estática dentro del archivo `src/views/static/preguntas-teoricas.html.ejs`. Dentro de esta vista encontrará una etiqueta `<p></p>` vacía bajo cada pregunta. **Usted debe utilizar estas etiquetas para ingresar sus respuestas**. Para visualizar su implementación como parte de la vista HTML, puede refrescar la ruta `/preguntas-teoricas` las veces que desee.

Consideraciones

- Sólo se aceptarán respuestas dentro de las etiquetas de párrafo especificadas
- Se le pide que intente no extenderse más de 3 líneas considerando un ancho de pantalla normal (puede tomar como referencia un ancho de `920px`)
- Debe rellenar sólo las etiquetas asociadas a las preguntas que usted seleccionó
- Si incluye más de 5 respuestas, sólo se corregirán las 5 primeras. Por lo tanto, tenga cuidado de no agregar una respuesta a una pregunta que no seleccionó
- Todas las preguntas tienen el mismo puntaje

Parte II: Landing estático [1.5 pts]

En esta parte usted debe implementar la landing page de la plataforma. Si ingresa a la ruta raíz de la aplicación, podrá ver que, además de un pequeño layout que incluye un header (el cual es compartido en toda la aplicación), la vista asociada a esta ruta está en blanco.

A continuación encontrará dos wireframes de posibles landing pages. Usted debe seleccionar una de las dos opciones e implementar una vista basada en el wireframe, utilizando solamente HTML y CSS.

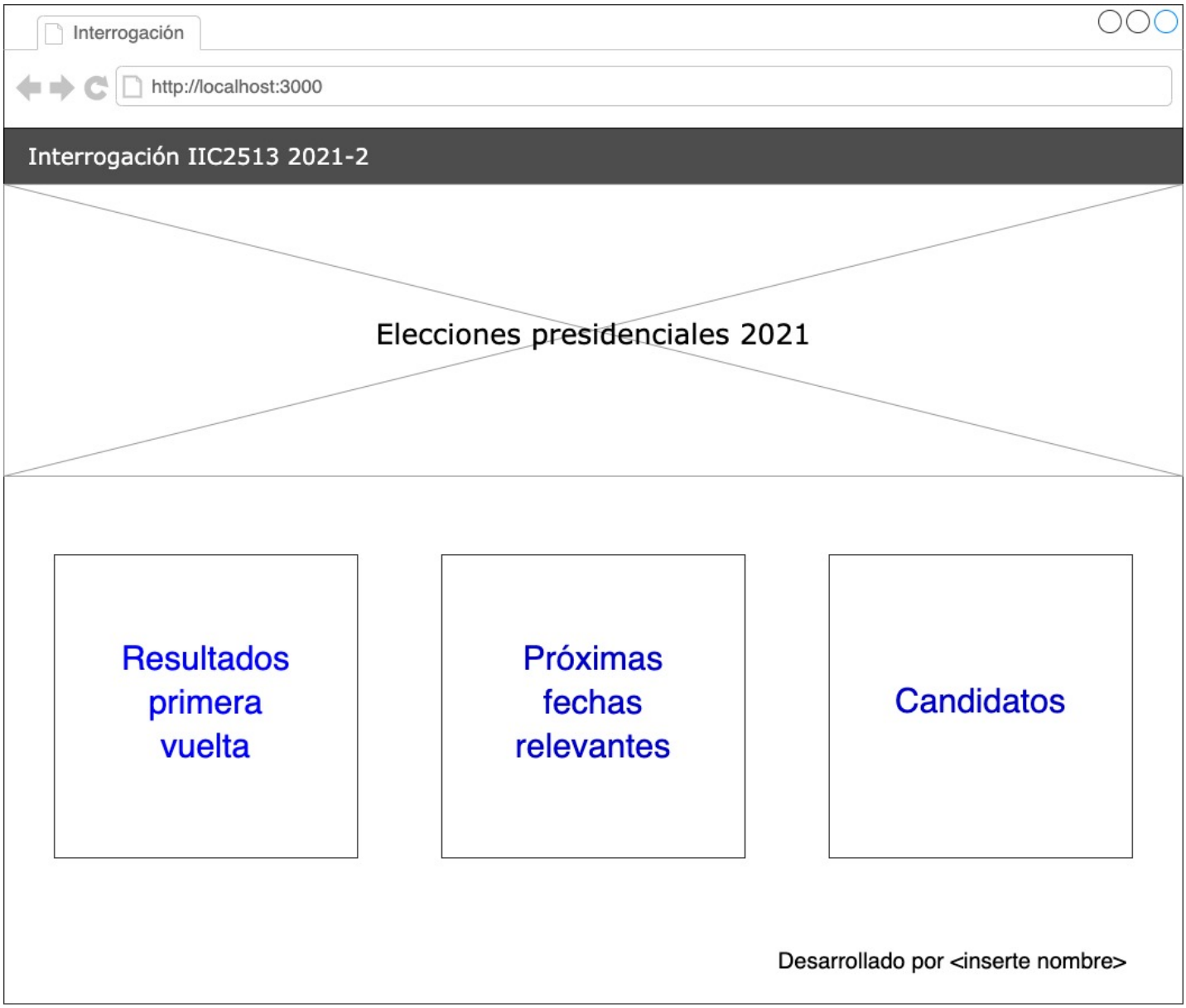
Wireframe 1

Solución

```
<div class="first-initial-box">
  <section>
    <h1>Elecciones presidenciales 2021</h1>
    <p>El pasado domingo 21 de noviembre se llevó a cabo el proceso para elegir un nuevo presidente, parlamentarios y COREs.</p>
    <span>Habrá segunda vuelta</span>
  </section>
  <!-- Fuente: https://www.pauta.cl/pauta/site/artic/20210127/imag/foto_0000015920210127191745/xCandidatos-v13.jpg.pagespeed.ic.HtQSIGprd4.jpg -->
  <img src="<%= assetPath('candidatos-first.jpg') %>" />
</div>

<div class="first-secondary-box">
  <p class="paragraph">
    Los candidatos José Antonio Kast y Gabriel Boric obtuvieron la mayor cantidad de sufragios.
    Por lo tanto, disputarán el sillón presidencial el próximo 19 de diciembre.
    Si quieres conocer más detalles, <a href="#">haz click aquí</a>.
  </p>
</div>

<footer>
  <p>Desarrollado por <strong>Sebastián Vicencio</strong></p>
</footer>
```

```
.first-initial-box {
  display: flex;
  border: 1px solid $main-color;

  section {
    display: flex;
    flex-direction: column;
    padding: 25px;

    h1 {
      font-size: 2.5rem;
      margin: 0.5rem 0 0;
    }

    p {
      font-size: 1.2rem;
    }

    span:last-child {
      order: -1;
    }
  }

  img {
    width: 60%;
    height: auto;
  }
}

.first-secondary-box {
  padding: 50px 0;

  p {
    font-size: 1.5rem;
    text-align: center;
  }
}

.paragraph {
  a {
    color: $app-blue; // Or other color to differentiate
    font-weight: bold; // Only to highlight (not required)
  }
}

footer {
  text-align: right;
}
```

Wireframe 2



Solución

```
<div class="second-initial-box">
  <h1>Elecciones presidenciales 2021</h1>
  <!-- https://www.principal.cl/sites/default/files/styles/755_250/public/header_paper_elecciones.png?itok=ura0c4FB -->
  <img src="<%= assetPath('elecciones-second.png') %>" />
</div>

<div class="second-secondary-box">
  <div class="second-content">
    <p>
      <a href="#">Resultados primera vuelta</a>
    </p>
  </div>
  <div class="second-content">
    <p>
      <a href="#">Próximas fechas relevantes</a>
    </p>
  </div>
  <div class="second-content">
    <p>
      <a href="<%= paths.candidates %>">Candidatos</a>
    </p>
  </div>
</div>

<footer class="second-footer">
  <p>Desarrollado por <strong>Sebastián Vicencio</strong></p>
</footer>
```

```
.second-initial-box {
  position: relative;

  h1 {
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
    color: white;
    font-size: 2.5rem;
  }

  img {
    width: 100%;
    height: auto;
  }
}

.second-secondary-box {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 50px;
  padding: 50px;

  .second-content {
    padding: 20% 10%;
    border: 1px solid $main-color;
    display: flex;
    align-items: center;
    justify-content: center;

    p {
      text-align: center;
      font-size: 2rem;
    }
  }
}

.second-footer {
  padding: 0 50px;
}

footer {
  text-align: right;
}
```

Consideraciones para ambas opciones

- Puede utilizar cualquier estrategia que conozca para disposición de elementos (Flex, Grid, float, etc)
- No es necesario que utilice etiquetas HTML semánticas
- Lo que se muestra en azul son enlaces. Puede utilizar el enlace que desee o colocar un placeholder como "#"
- Las cajas con dos líneas diagonales son imágenes. Puede seleccionar la imagen que sea de su preferencia. Para asegurarse de que sea visualizada en la corrección, puede agregar la imagen al proyecto dentro de `src/assets/images` (no olvide en ese caso agregar la referencia a `src/assets/js/assets.js`)
- No debe utilizar estilos inlines. Si lo hace, tendrá un descuento de 0.3 pts
- La sintaxis del HTML implementado debe estar correcta. Cualquier error significará un descuento de 0.2 pts

Pauta de evaluación

Wireframe 1

- **[0.30 pts]** Existe una caja inicial que incluye una sub-caja y una imagen dispuestos de forma horizontal (uno al lado del otro)
- **[0.20 pts]** Ambos elementos dentro de la caja inicial tienen la misma altura y un borde visible
- **[0.30 pts]** La sub-caja del primer punto incluye un epígrafe, título y párrafo dispuestos de forma vertical, con los textos especificados en el wireframe
- **[0.20 pts]** Existe una sección bajo la caja inicial que incluye un párrafo como el especificado en el wireframe, con el texto centrado
- **[0.25 pts]** El párrafo incluye un enlace que se puede visualizar con un color diferente, y que apunta ya sea a una URL o a "`#`"
- **[0.25 pts]** Existe un footer con el texto alineado a la derecha, como el especificado en el wireframe
  - Idealmente la solución debería incluir el nombre del alumno en reemplazo del texto "`<inserte nombre>`", pero no es obligatorio

Descuentos

- **[-0.30 pts]** Uso de estilos inline
- **[-0.20 pts]** Documento HTML con mala sintaxis. Validar con herramienta de W3C, opción "Validate by Direct Input". Se pueden ignorar "warnings" y errores que hagan referencia a elementos no relevantes para el caso (por ejemplo, la falta del atributo "alt" para las imágenes). Tener cuidado de reemplazar eventuales llamadas a `assetPath` con la correspondiente URL

Wireframe 2

- **[0.20 pts]** Existe una caja inicial que incluye una imagen y un título, como el especificado en el wireframe
- **[0.20 pts]** La imagen de la caja inicial abarca todo el ancho del contenedor (clase `.container`) (o al menos se aprecia claramente una diferencia de ancho con el resto del documento)
- **[0.30 pts]** El título de la caja inicial está posicionado sobre la imagen, justo en el centro de esta (tanto vertical como horizontalmente), y debe poder visualizarse bien (contraste de colores)
- **[0.30 pts]** Existe una sección que incluye 3 cajas dispuestas de forma horizontal (una al lado de la otra), con el texto especificado en el wireframe

- **[0.25 pts]** El texto de cada caja corresponde a un enlace que se puede visualizar con un color diferente, y que apunta ya sea a una URL o a "`#`"
- **[0.25 pts]** Existe un footer con el texto alineado a la derecha, como el especificado en el wireframe
    - Idealmente la solución debería incluir el nombre del alumno en reemplazo del texto "`<inserte nombre>`", pero no es obligatorio

Descuentos

- **[-0.30 pts]** Uso de estilos inline
- **[-0.20 pts]** Documento HTML con mala sintaxis. Validar con herramienta de W3C, opción "Validate by Direct Input". Se pueden ignorar "warnings" y errores que hagan referencia a elementos no relevantes para el caso (por ejemplo, la falta del atributo "alt" para las imágenes). Tener cuidado de reemplazar eventuales llamadas a `assetPath` con la correspondiente URL

Consideraciones

- Queda a criterio del ayudante corrector asignar puntaje total o parcial de acuerdo a completitud del requerimiento
- No es necesario utilizar `assetPath`. Las soluciones pueden incluir una imagen de un tercero, sin embargo, es responsabilidad del alumno que se pueda visualizar al momento de la corrección. En caso de no poder visualizarse, no debe asignarse todo el puntaje en el ítem que evalúe el uso de la imagen
- No es necesario que la vista sea responsive. Basta que se visualice como en el wireframe con un tamaño común de pantalla desktop
- La solución planteada es una de las posibilidades. Existen diferentes formas de lograr el mismo resultado, por ejemplo, utilizando Flexbox, Grid o float, y todas son válidas

Parte III: API en koa [3.5 pts]

En esta última parte usted debe implementar 3 endpoints desde cero, detallados en las siguientes secciones.

Lista de candidatos [1.0 pt]

Debe implementar un endpoint que permita listar los candidatos presidenciales. La especificación del endpoint es la siguiente:

- Method: `GET`
- Path: `/api/candidates`
- Response
    - Status code: `200`
    - Content-Type: `application/json`

El cuerpo del response no está especificado intencionalmente en este documento, ya que usted debe deducirlo a partir de una implementación existente en la aplicación. Esto se detalla a continuación.

Si usted ingresa a la ruta `/candidates` en el browser podrá ver una vista que muestra inicialmente que no hay información disponible para candidatos. Sin embargo, al revisar el detalle de la vista EJS asociada, no encontrará lo mismo que se visualiza en el browser, sino que sólo verá un elemento HTML vacío con id `root`. El resto está implementado mediante una pequeña aplicación frontend montada sobre este elemento del DOM, cuya implementación se encuentra dentro de `src/assets/js/app.jsx`.

Para poder conocer el cuerpo del response del endpoint para listar candidatos, deberá revisar el código frontend mencionado en el párrafo anterior. Ahí encontrará que se consume el endpoint de la API asociado a esta lista, y se utilizan ciertos campos del response para renderizar el resultado. **Esos campos son los mínimos que usted debe incluir el cuerpo del response**. Tenga en consideración que uno de los campos hace referencia a la edad de los candidatos. Puede utilizar JavaScript nativo o bien la librería `date-fns` (ya instalada) para obtener este campo.

Nota: La aplicación frontend hace uso del DOM con la API nativa del browser (no utiliza React en absoluto).

Solución

```
const differenceInYears = require('date-fns/differenceInYears');

// ...

function buildCandidatePayload(candidate) {
  const {
    id, name, party, birthdate, description, photo, votes,
  } = candidate;
  return {
    id,
    name,
    party,
    years: differenceInYears(new Date(), new Date(birthdate)),
    description,
    photo,
    votes,
  };
}

router.get('api.candidates.index', '/', async (ctx) => {
  const candidates = await ctx.orm.candidate.findAll();
  ctx.body = candidates.map(buildCandidatePayload);
});
```

Pauta de evaluación

- **[0.25 pts]** Definición de nuevo endpoint con método HTTP `GET` y path `/candidates`
- **[0.25 pts]** Consulta dentro del endpoint para obtener datos solicitados del modelo asociado
- **[0.20 pts]** Construcción del body a partir de la lista de candidatos (iterar sobre lista o alguna otra estrategia)
- **[0.30 pts]** Inclusión de los 6 campos requeridos por el frontend
    - Asignar **0.05 pts** por cada campo: `name`, `party`, `years`, `description`, `photo`, `votes`.

Programas de candidatos [1.0 pt]

Debe implementar un endpoint que permita obtener los principales puntos del programa de un candidato en particular. La especificación del endpoint es la siguiente:

- Method: `GET`

- Path: `/api/candidates/:id/proposals`
- Response
  - Status code: `200`
  - Content-Type: `application/json`

Para el cuerpo del response, tendrá que elegir entre dos alternativas, que presentan la información del programa de gobierno de forma diferente, y las cuales se detallan a continuación.

Opción 1

```
[
  {
    "topic1": "content",
    "candidateLastName": "Last name of candidate",
    "createdAt": "November 25th, 2021"
  },
  {
    "topic2": "content",
    "candidateLastName": "Last name of candidate",
    "createdAt": "November 25th, 2021"
  }
]
```

Cada elemento del response debe tener sólo esas keys (ni más ni menos). Además, puede suponer que "createdAt" es la fecha en que se creó esa propuesta de gobierno (aunque sea el "createdAt" predeterminado de Sequelize). Este campo tiene que tener el formato "November 25th, 2021". El proyecto tiene instalado el package `date-fns` , que puede ser utilizado para conseguir ese formato, por lo que se sugiere buscar en la documentación de este package. Puede suponer que el candidato siempre tendrá su apellido al final e ignorar apellidos compuestos.

Opción 2

```
{
  "topic1-slug": {
    "topic": "topic value",
    "content": "content value",
    "candidate": "Name of candidate"
  },
  "topic2-slug": {
    "topic": "topic value",
    "content": "content value",
    "candidate": "Name of candidate"
  }
}
```

Donde `topic-slug` es el `topic` procesado de tal forma que tenga sólo minúsculas, ninguna letra con tilde y que los espacios hayan sido reemplazados por `-` . Puede suponer que un tópico nunca vendrá que otros caracteres "extraños" fuera de los mencionados. Por ejemplo, si el tópico es "Planificación urbana", el topic slug sería "planificacion-urbana".

Errores

- En caso de no existir el candidato, el endpoint debe retornar un**status code 404** (Not Found), con un pequeño mensaje descriptivo a su elección

Solución opción 1

```
const format = require('date-fns/format');

// ...

router.param('id', async (id, ctx, next) => {
  const candidate = await ctx.orm.candidate.findByPk(id);
  if (!candidate) ctx.throw(404, "The candidate you are looking for doesn't exist");
  ctx.state.candidate = candidate;
  await next();
});

router.get('api.candidates.proposals', '/:id/proposals', async (ctx) => {
  const { candidate } = ctx.state;
  const proposals = await candidate.getProposals();

  const response = proposals.map(({ topic, content, createdAt }) => {
    const nameParts = candidate.name.split(' ');
    return {
      [topic]: content,
      candidateLastName: nameParts[nameParts.length - 1],
      createdAt: format(createdAt, 'PPP'),
    };
  });

  ctx.body = response;
});
```

Pauta de evaluación opción 1

- **[0.15 pts]** Definición de nuevo endpoint con método HTTP `GET` y path `/candidates/:id:/proposals`
- **[0.15 pts]** Consulta dentro del endpoint para obtener el candidato dado por el parámetro `id`
- **[0.15 pts]** Consulta dentro del endpoint para obtener el listado de propuestas programáticas de un candidato específico
- **[0.45 pts]** El body response es un array que incluye un elemento por tópico de propuesta programática. Además, cada elemento de la lista es un objeto que tiene los campos `<nombre_topico>`, `candidateLastName` y `createdAt`, cuyos valores están detallados a continuación:
  - [0.20 pts] Incluye una key con el nombre del tópico, y que tiene como value el contenido asociado a ese tópico
  - [0.10 pts] Incluye el apellido del candidato bajo la llave `candidateLastName`
  - [0.15 pts] Incluye la fecha de creación de la propuesta bajo la llave `createdAt`, con el formato `'PPP'` que es posible utilizar con `date-fns`
- **[0.10 pts]** En caso de no existir el candidato, se retorna un error `404` con un mensaje descriptivo (no el default)

Solución opción 2

```
function generateSlug(str) {
  return str.toLowerCase()
    .replace(/á/, 'a')
    .replace(/é/, 'e')
    .replace(/í/, 'i')
    .replace(/ó/, 'o')
    .replace(/ú/, 'u')
    .replace(/\s+/g, '-');
}

router.param('id', async (id, ctx, next) => {
  const candidate = await ctx.orm.candidate.findByPk(id);
  if (!candidate) ctx.throw(404, "The candidate you are looking for doesn't exist");
  ctx.state.candidate = candidate;
  await next();
});

router.get('api.candidates.proposals', '/:id/proposals', async (ctx) => {
  const { candidate } = ctx.state;
  const proposals = await candidate.getProposals();

  const response = proposals.reduce((acc, { topic, content }) => ({
    ...acc,
    [generateSlug(topic)]: {
      topic,
      content,
      candidate: candidate.name,
    },
  }), {});

  ctx.body = response;
});
```

Pauta de evaluación opción 2

- **[0.15 pts]** Definición de nuevo endpoint con método HTTP `GET` y path `/candidates/:id:/proposals`
- **[0.15 pts]** Consulta dentro del endpoint para obtener el candidato dado por el parámetro `id`
- **[0.15 pts]** Consulta dentro del endpoint para obtener el listado de propuestas programáticas de un candidato específico
  - Puede ser tanto explícitamente con una nueva consulta, como también dentro de la consulta anterior, utilizando `include`
- **[0.30 pts]** El body response es un objeto que tiene tantas keys como propuestas programáticas, y cada key tiene como value otro objeto que incluye los campos `topic`, `content` y `candidate`, cuyos valores están detallados a continuación:
  - [0.10 pts] Incluye slugs como keys (independiente de si están bien construidos)
  - [0.05 pts] Incluye el tópico de la propuesta programática bajo la llave `topic`
  - [0.05 pts] Incluye el contenido de la propuesta programática bajo la llave `content`
  - [0.10 pts] Incluye el nombre del candidato bajo la llave `candidate`
- **[0.15 pts]** Los slugs de cada tópico están construidos acorde a lo especificado en el enunciado (sólo minúsculas, ninguna letra con tilde y espacios reemplazados por `-`)
- **[0.10 pts]** En caso de no existir el candidato, se retorna un error `404` con un mensaje descriptivo (no el default)

Acción sobre candidatos [1.0 pt]

El último endpoint que debe implementar será nuevamente seleccionado por usted entre dos opciones. En ambos casos, eso sí, el endpoint estará protegido y sólo podrá ser accedido por usuarios que posean un JSON Web Token (JWT) válido, el cual **debe ser incluido en cada request** que se realice.

Lo anterior significa que un usuario deberá inicialmente iniciar sesión para poder acceder a este endpoint. El endpoint de autenticación ya se encuentra implementado, y su especificación es:

- Method: **`POST`**
- Path: **`/api/auth`**
- Content-Type: **`application/json`**
- Body parameters:

```
{
  "email": "user_email",
  "password": "user_password"
}
```

- Response
  - Status code: `201`
  - Content-Type: `application/json`

- Body:

```
{
  "access_token": "jwt_token",
  "token_type": "Bearer"
}
```

IMPORTANTE: no olvide agregar la variable de ambiente **JWT_SECRET** para que la generación del token funcione sin problemas

En las seeds del proyecto existen usuarios válidos para la aplicación, cuyas credenciales se sugiere utilizar para pruebas. Uno de esos usuarios tiene las siguientes credenciales:

- E-mail: user@example.org
- Password: hola.123

A continuación, entonces encontrará el detalle de ambas opciones de endpoints. Recuerde que debe seleccionar uno solo e implementarlo.

Votar por candidato

Este endpoint permite votar por un candidato en particular. La cantidad de votos se encuentra reflejada en el campo `votes` del modelo `candidate`. Usted debe incrementar en 1 el valor de este campo por cada vez que se llame a este endpoint. La especificación del endpoint es la siguiente:

- Method: **`POST`**
- Path: **`/api/candidates/:id/votes`**
- Content-Type: **`application/json`**
- Body parameters: no requiere
- Response
  - Status code: `201`
  - Content-Type: `application/json`
  - Body: debe incluir la información del candidato utilizando los mismos campos incluidos en un elemento del endpoint para listar candidatos

Errores

- En caso de no incluir información de autenticación o que esta sea inválida, el endpoint debe retornar un **status code 401** (Unauthorized), con un pequeño mensaje descriptivo a su elección
- En caso de no existir el candidato, el endpoint debe retornar un **status code 404** (Not Found), con un pequeño mensaje descriptivo a su elección

Solución

```
require('dotenv').config();
const jwt = require('koa-jwt');

// ...

router.use(jwt({ secret: process.env.JWT_SECRET, key: 'authData' }));

router.post('api.candidates.votes', '/:id/votes', async (ctx) => {
  const { candidate } = ctx.state;
  await candidate.update({ votes: candidate.votes + 1 });
  ctx.status = 201;
  ctx.body = buildCandidatePayload(candidate);
});
```

Pauta de evaluación

- **[0.15 pts]** Definición de nuevo endpoint con método HTTP `POST` y path `/candidates/:id:/votes`
- **[0.15 pts]** Consulta dentro del endpoint para obtener el candidato dado por el parámetro `id`
- **[0.25 pts]** Actualización de la cantidad de votos del candidato hecha correctamente (incrementar valor en 1)
- **[0.10 pts]** Response incluye status code `201`
- **[0.15 pts]** Response incluye cuerpo que consiste en un objeto con la misma estructura que la de cada elemento del endpoint de lista de candidatos (es decir, incluye los campos `name`, `party`, `years`, `description`, `photo`, `votes`, construidos correctamente)
- **[0.10 pts]** En caso de no incluir información de autenticación o que esta sea inválida, se retorna un error `401`, con un pequeño mensaje descriptivo (puede ser el default)
- **[0.10 pts]** En caso de no existir el candidato, se retorna un error `404` con un mensaje descriptivo (no el default)

Modificar descripción de candidato

Este endpoint permite modificar la descripción asociada a un candidato en particular. Debe sólo permitir modificar este campo y no otros. La especificación del endpoint es la siguiente:

- Method: **`PATCH`**
- Path: **`/api/candidates/:id`**
- Content-Type: **`application/json`**

- Body parameters

```
{
  "description": "Modified description"
}
```

- Response
  - Status code: `200`
  - Content-Type: `application/json`
  - Body: debe incluir la información del candidato utilizando los mismos campos incluidos en un elemento del endpoint para listar candidatos

Errores

- En caso de no incluir información de autenticación o que esta sea inválida, el endpoint debe retornar un **status code 401** (Unauthorized), con un pequeño mensaje descriptivo a su elección
- En caso de no existir el candidato, el endpoint debe retornar un **status code 404** (Not Found), con un pequeño mensaje descriptivo a su elección
- En caso de ingresar una descripción vacía o nula, el endpoint debe retornar un **status code 422** (Unprocessable entity), con un pequeño mensaje asociado a la falta de este campo
- En caso de ingresar algún campo diferente a `description`, el endpoint debe retornar un **status code 400** (Bad request), con un pequeño mensaje explicando que sólo se permite modificar la descripción

Solución

```
 require('dotenv').config();
const jwt = require('koa-jwt');

// ...

router.use(jwt({ secret: process.env.JWT_SECRET, key: 'authData' }));

router.patch('api.candidates.update', '/:id', async (ctx) => {
  const { candidate } = ctx.state;
  const { description, ...restBody } = ctx.request.body;
  if (Object.keys(restBody).length) ctx.throw(400, 'Only description can be updated');
  try {
    await candidate.update({ description });
  } catch (ValidationError) {
    ctx.throw(422, ValidationError.message);
  }

  ctx.body = buildCandidatePayload(candidate);
});
```

Pauta de evaluación

- **[0.15 pts]** Definición de nuevo endpoint con método HTTP `PATCH` y path `/candidates/:id`
- **[0.15 pts]** Consulta dentro del endpoint para obtener el candidato dado por el parámetro `id`
- **[0.15 pts]** Actualización de la descripción del candidato hecha correctamente, recibiendo el nuevo valor como parte del cuerpo del request, bajo la key `description`
- **[0.15 pts]** Response incluye cuerpo que consiste en un objeto con la misma estructura que la de cada elemento del endpoint de lista de candidatos (es decir, incluye los campos `name`, `party`, `years`, `description`, `photo`, `votes`, construidos correctamente)
- **[0.10 pts]** En caso de no incluir información de autenticación o que esta sea inválida, se retorna un error `401`, con un pequeño mensaje descriptivo (puede ser el default)
- **[0.10 pts]** En caso de no existir el candidato, se retorna un error `404` con un mensaje descriptivo (no el default)
- **[0.10 pts]** En caso de ingresar una descripción vacía o nula, se retorna un error `422` con un pequeño mensaje asociado a la falta de este campo
- **[0.10 pts]** En caso de ingresar algún campo diferente a `description`, se retorna un error `400` con un pequeño mensaje explicando que sólo se permite modificar la descripción

Calidad de software: análisis estático de código [0.5 pt]

El resultado final debe permitir ejecutar ESLint sin que arroje errores.

Pauta de evaluación

- **[0.5 pts]** Linter ejecuta exitosamente, sin arrojar errores ni warnings

Consideraciones generales

- El proyecto cuenta con todos los packages necesarios para desarrollar la Interrogación. De todas formas, si considera que algún package le ayudará, puede incluir un archivo README dentro de la carpeta src indicando cualquier package extra utilizado, y cualquier instrucción que el ayudante requiera conocer para ejecutar su solución
- En general, el enunciado es bastante claro en cuanto a qué archivos usted debe modificar. Sin embargo, usted es libre de hacer modificaciones a otros archivos o agregar nuevos si lo estima conveniente, siempre que estén dentro de la carpeta "src" del proyecto, exceptuando los relacionados a los siguientes 2 puntos
- Usted no debe modificar el código JavaScript client-side. Sólo lo necesita utilizar para identificar algunas especificaciones de los endpoints de la API, y para testear que su implementación está correcta. Si lo modifica para ajustar su implementación de la API, este último ítem no será válido
- Usted no debe modificar los modelos de la aplicación. Si lo hace, las implementaciones que hagan uso de sus cambios no serán válidas
- Para agregar estilos, puede utilizar el archivo en la ruta `src/assets/styles/app.scss`
- No se evaluará lo que no sea posible probar, por lo tanto, asegúrese de que su aplicación corra sin problemas (puede instalarla "desde cero" para asegurarse)

Aspectos administrativos

Forma de entrega

Para entregar la solución de su Interrogación, debe subir **un solo archivo .zip** al formulario que se publicará en un anuncio. Este archivo debe contener sólo lo siguiente: carpeta `src` del repositorio clonado y que contenga la implementación de todas las respuestas.

El nombre del archivo .zip debe seguir el siguiente formato:

```
apellidoPaterno_Nombre_numeroAlumno.zip
```

Por ejemplo, el alumno Benjamín Moisés Retamal Palacios entregará un archivo llamado: `Retamal_Benjamin_13191510.zip`

Asegúrese de que el archivo .zip contiene todos los archivos de la interrogación. Para esto se le sugiere descomprimirlo y revisar el contenido antes de enviarlo en el formulario. Lo que se reciba es lo que se corregirá, **sin excepciones**.

Política de atrasos

Según lo especificado en el programa del curso, **el primer día de atraso descontará un 25% del puntaje total obtenido**. Un día de retraso comienza al segundo siguiente del plazo establecido (como el plazo es a las 22:00:00, entonces entregar a las 22:00:01 ya se considera un día de atraso). **Al segundo día de atraso, se califica la evaluación con nota 1.0**.

Consultas

1. Se responderán consultas (dudas estrictamente de enunciado) en la primera hora del módulo de clases (de 10:00 hrs - 11:00 hrs), en la misma sesión Zoom de las cátedras. De este modo, usted alcanza a leer el enunciado antes y llegar a plantear dudas. De todas maneras, se publicará un resumen de las dudas en una issue del repositorio `syllabus`.
2. **No habrá preguntas por medio de issues**, por lo que se sugiere encarecidamente asistir a la sesión por Zoom y/o revisar el video asociado.

¡Éxito!