INSTITUTO DE EDUCACIÓN SUPERIOR PROFESIONAL CIBERTEC

PROYECTO DE CURSO

“EXPENSES TRACKER”

SEDE: AREQUIPA

CARRERA: COMPUTACIÓN E INFORMÁTICA

INTEGRANTES:

CALDERON CORNEJO, ANGELO
HUANACO QUISPE, JUAN DIEGO
MAMANI USCAMAYTA, AGUSTIN DAVID
PEREDO SUCSO, MOISES KEVIN
PINEDA ACEITUNO, EVER HUGO

DESARROLLO DE APLICACIONES WEB I

DOCENTE A CARGO: JESUS ANTONIO ALPACA RENDON

5TO CICLO

PROYECTO GRUPAL

PERU / 2023

ÍNDICE

RESUMEN 3

INTRODUCCIÓN 3

DIAGNÓSTICO 3

JUSTIFICACIÓN DEL PROYECTO 4

OBJETIVOS 4

DEFINICIÓN Y ALCANCE 4

PRODUCTOS Y ENTREGABLES 5

CONCLUSIONES 10

RECOMENDACIONES 10

GLOSARIO 10

BIBLIOGRAFÍA 10

Resumen

En el presente documento se expone el desarrollo de una aplicación web, teniendo en consideración la importancia de saber llevar un control del flujo del dinero para una buena estabilidad económica, se ha creado “Expenses Tracker”, el cual es un proyecto desarrollado por varios integrantes que cuenta con diversas funciones para ayudar a llevar un registro de los ingresos y gastos de los usuarios registrados.

El proyecto apunta a resolver el problema del manejo de finanzas, problema que aqueja no solo a adultos sino a también a los jóvenes que tomar el rol de sus propios consultores cuando se trata de la gestión del dinero.

Introducción

El control de los gastos es un factor clave en la distribución de cantidad adecuada de dinero destinado a los gastos de cada persona, puesto que: *“Es a través de este proceso que serás capaz de identificar cuánto dinero estás gastando y en qué productos y/o servicios, y ello te dará además la posibilidad de entender cómo organizar los gastos de la casa.”* (The InvestorU, s.f.); El correcto manejo de las finanzas personales no es exclusivo de adultos, también los jóvenes deben tengan control sobre sus gastos y ahorros. “Expenses Tracker” nace con la intención de brindar una solución alterna a las típicas plantillas de Excel que se pueden encontrar en Internet; contando con una interfaz sencilla de usar para todo público; con las herramientas necesarias para el control del presupuesto, los gastos e ingresos; incluyendo registros de toda operación financiera y la capacidad de exportar estos datos de forma sencilla.

Diagnóstico

Para garantizar una vida útil del proyecto se hizo un análisis del tipo SEPTE para conocer los factores externos que pudiesen afectarlo.

Por el lado económico, lo cual es el principal punto de desarrollo que busca el proyecto, tenemos proyectos similares, aunque enfocados en los dispositivos móviles, han tenido resultados generalmente positivos brindados por los mismos cibernautas, casos como: “Mint: Budget & Track Bills”[1], “Toshl Finanzas - presupuesto”[2], “Finerio: Finanzas Personales”[3], entre otros; reflejan casos de éxito en su mayoría.

En lo social, se sabe que en los últimos 5 años los jóvenes han adoptado una serie de conductas negativas[4], tales como: Bajo conocimiento financiero, no estar contento con sus finanzas personales, etc.; este proyecto cuenta con el potencial para generar un impacto positivo en tales aspectos.

Y por el lado tecnológico, desarrollar una aplicación web está actualizado a los estándares de consumo de información de la población, ganando ventaja de presencia en el ciberespacio para el proyecto.[5]

[1] App Mint: https://play.google.com/store/apps/details?id=com.mint&gl=US
[2] App Toshl: https://play.google.com/store/apps/details?id=com.thirdframestudios.android.expensoor&gl=US
[3] App Finerio: https://play.google.com/store/apps/details?id=mx.finerio&gl=US
[4] Información consultada en: https://peru.com/actualidad/economia-y-finanzas/jovenes-y-sus-finanzas-cual-su-comportamiento-economico-noticia-501056/?ref=pcom
[5] Datos extraídos de: https://revistaempresarial.com/tecnologia/la-importancia-de-las-aplicaciones-web-y-moviles-en-el-exito-empresarial/

Justificación del Proyecto

En base a las investigaciones "Bancarización 2021" y "Generaciones en el Perú 2020" realizadas por IPSOS hacia la población peruana, sabemos que 4 de 10 personas afirman haber ahorrado menos de lo que tenía pensando, además el 63% de los encuestados presentan deudas por préstamos con entidades financieras. Se puede categorizar como regular la eficiencia del peruano para administrar los ahorros. Por ello propongo la creación de una herramienta para incrementar la eficiencia del ahorro en las personas. Se hará uso de una aplicación web, puesto que, *"Cualquier persona puede hoy en día acceder a las aplicacionesweb por medio de un navegador moderno, presente prácticamente en cualquier computador […] Las posibilidades de masificación del producto o servicio son enormes y esto se complementa además con nuevas y mejores funcionalidades de accesibilidad, las cuales permiten a personas con alguna discapacidad hacer uso de estos sistemas."* (Gutiérrez Tiuso, s.f.).

Objetivos

- **Objetivo 1**
  Lograr que, en el plazo de un mes desde su implementación, más del 50% de los usuarios mejoren su control financiero, validando con la métrica del capital[6].

- **Objetivo 2**
  Conseguir brindar, para el día de la implementación, una interfaz sencilla de usar y comprender para más del 70% de los usuarios, teniendo como métrica la opinión de los testers[7] para pulir la interfaz.

Definición y Alcance

- **Definición**
  La aplicación web será desarrollado con las tecnologías de Spring, JPA, Hibernate y Jasper Reports y a la vez debe contar con los siguientes módulos:

    - Dashboard o Vista General
    - Listado de registros (básico y avanzado)
    - Métricas de rendimiento
    - Ajustes de la cuenta (información personal y preferencias)

- **Alcance**
  Desarrollo de un sistema basado en tecnologías web para el servicio de la herramienta Expenses Tracker.

    - Entregables:
      El producto resultante se alojará en un servidor web gratuito para su exposición donde estará disponible al público durante el tiempo de 2 semanas, con ello se brindará la documentación necesaria para su correcto uso y mantenimiento.

[6] El programa monitorea su capital diariamente, si se llega a fin de mes con un capital positivo, entonces puede decirse que se ha conseguido un buen manejo de las finanzas.
[7] Se aplicarán encuestas al grupo de testers de la versión beta de la web app para saber en qué puede mejorarse la interfaz antes de ser implementada oficialmente.

- Características de los entregables:
  El programa estará listo para ser subido al servidor cuando todos los módulos sean sido terminados y se haya completado el proceso de pruebas a cargo del desarrollador hacia un grupo de testers.

- Criterios de aceptación:
  El proyecto se considerará finalizado cuando se cuente con el acabado de la documentación, la aprobación del grupo de testers consultado y del docente a cargo.

- Restricciones:
  El proyecto está sometido al tiempo del estudiante desarrollador y la disponibilidad de un servicio de alojamiento web gratuito para cumplir con el lapso de exposición al público que se propone (2 semanas).

Productos y Entregables

Repositorio del Proyecto: https://github.com/juanhuanaco/expensesTrackerSpring

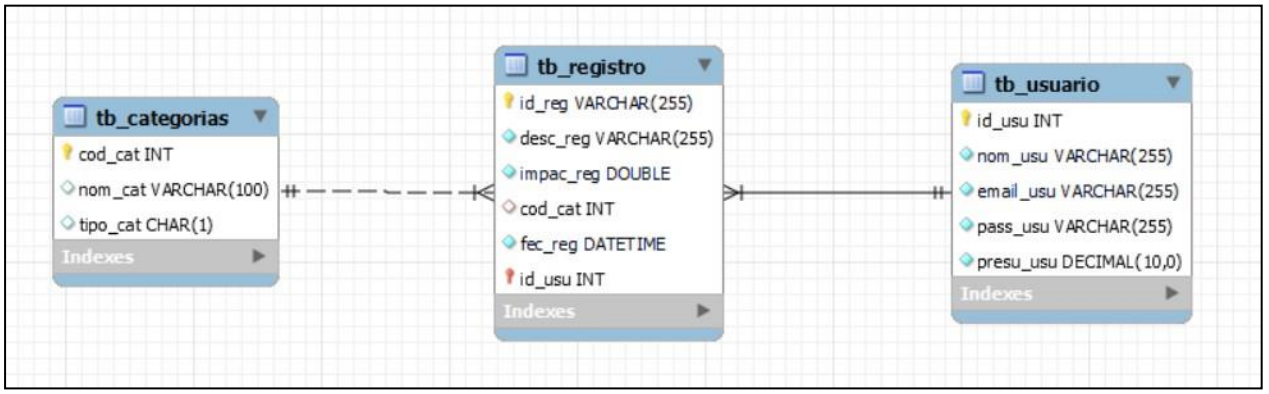

Imagen 1: Diagrama de la Base de Datos de Expenses Tracker – Fase Inicial

Models

```java
@Entity
public class Categoria {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "cod_cat")
    private int codigo;

    @Column(name = "nom_cat")
    private String nombre;

    @Column(name = "tipo_cat")
    private String tipo;
```

Models: Categoría

```java
@Entity
public class Registro {
    @Id
    @Column(name = "id_reg")
    private String id;

    @Column(name = "desc_reg", nullable = false)
    private String descripcion;

    @Column(name = "impac_reg", nullable = false)
    private double impacto;

    @Column(name = "fec_reg", nullable = false)
    private Date fecha;

    @ManyToOne
    @JoinColumn(name = "cod_cat")
    private Categoria categoria;

    @ManyToOne
    @JoinColumn(name = "id_usu")
    private Usuario usuario;
```

Models: Registro

```java
@Entity
public class Usuario implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_usu")
    private int id;

    @Column(name = "nom_usu")
    private String nombre;

    @Column(name = "email_usu")
    private String email;

    @Column(name = "pass_usu")
    private String password;

    @Column(name = "presu_usu")
    private double presupuesto;

    @OneToMany(mappedBy = "usuario")
    List<Registro> registros;

    public List<Registro> getRegistros() {
      return registros;
    }

    public void setRegistros(List<Registro> registros) {
      this.registros = registros;
    }
```

Models: Usuario

Controllers

```java
@RestController
@RequestMapping("/categorias")
public class CategoriaController {

    private final CategoriaService categoriaService;

    @Autowired
    public CategoriaController(CategoriaService categoriaService) {
        this.categoriaService = categoriaService;
    }

    @GetMapping
    public ResponseEntity<List<Categoria>> obtenerCategorias() {
        List<Categoria> categorias = categoriaService.obtenerCategorias();
        return ResponseEntity.ok(categorias);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Categoria> obtenerCategoriaPorId(@PathVariable int id) {
        Categoria categoria = categoriaService.obtenerCategoriaPorId(id);
        if (categoria != null) {
            return ResponseEntity.ok(categoria);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PostMapping
    public ResponseEntity<Categoria> crearCategoria(@RequestBody Categoria categoria) {
        Categoria nuevaCategoria = categoriaService.crearCategoria(categoria);
        return ResponseEntity.status(HttpStatus.CREATED).body(nuevaCategoria);
    }
```

```
    @PutMapping("/{id}")
    public ResponseEntity<Categoria> actualizarCategoria(@PathVariable int id, @RequestBody Categoria categoria) {
        Categoria categoriaExistente = categoriaService.obtenerCategoriaPorId(id);
        if (categoriaExistente != null) {
            categoria.setCodigo(id);
            Categoria categoriaActualizada = categoriaService.actualizarCategoria(categoria);
            return ResponseEntity.ok(categoriaActualizada);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarCategoria(@PathVariable int id) {
        boolean eliminado = categoriaService.eliminarCategoria(id);
        if (eliminado) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

Controllers: Categoría

```
@RestController
@RequestMapping("/registros")
public class RegistroController {

    private final RegistroService registroService;

    @Autowired
    public RegistroController(RegistroService registroService) {
        this.registroService = registroService;
    }

    @GetMapping
    public ResponseEntity<List<Registro>> obtenerRegistros() {
        List<Registro> registros = registroService.obtenerRegistros();
        return ResponseEntity.ok(registros);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Registro> obtenerRegistroPorId(@PathVariable String id) {
        Registro registro = registroService.obtenerRegistroPorId(id);
        if (registro != null) {
            return ResponseEntity.ok(registro);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PostMapping
    public ResponseEntity<Registro> crearRegistro(@RequestBody Registro registro) {
        Registro nuevoRegistro = registroService.crearRegistro(registro);
        return ResponseEntity.status(HttpStatus.CREATED).body(nuevoRegistro);
    }
```

```
    @PutMapping("/{id}")
    public ResponseEntity<Registro> actualizarRegistro(@PathVariable String id, @RequestBody Registro registro) {
        Registro registroActualizado = registroService.actualizarRegistro(id, registro);
        if (registroActualizado != null) {
            return ResponseEntity.ok(registroActualizado);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarRegistro(@PathVariable String id) {
        boolean eliminado = registroService.eliminarRegistro(id);
        if (eliminado) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

Controllers: Registro

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioController {

    private final UsuarioService usuarioService;

    @Autowired
    public UsuarioController(UsuarioService usuarioService) {
        this.usuarioService = usuarioService;
    }

    @GetMapping
    public ResponseEntity<List<Usuario>> obtenerUsuarios() {
        List<Usuario> usuarios = usuarioService.obtenerUsuarios();
        return ResponseEntity.ok(usuarios);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Usuario> obtenerUsuarioPorId(@PathVariable int id) {
        Usuario usuario = usuarioService.obtenerUsuarioPorId(id);
        if (usuario != null) {
            return ResponseEntity.ok(usuario);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PostMapping
    public ResponseEntity<Usuario> crearUsuario(@RequestBody Usuario usuario) {
        Usuario nuevoUsuario = usuarioService.crearUsuario(usuario);
        return ResponseEntity.status(HttpStatus.CREATED).body(nuevoUsuario);
    }
```

```
    @PutMapping("/{id}")
    public ResponseEntity<Usuario> actualizarUsuario(@PathVariable int id, @RequestBody Usuario usuario) {
        Usuario usuarioActualizado = usuarioService.actualizarUsuario(id, usuario);
        if (usuarioActualizado != null) {
            return ResponseEntity.ok(usuarioActualizado);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarUsuario(@PathVariable int id) {
        boolean eliminado = usuarioService.eliminarUsuario(id);
        if (eliminado) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

Controllers: Usuario

Repositories

```
@Repository
public interface IRegistroRepository extends JpaRepository<Registro, Long>  {

        //Lista todos los registros de un usuario en particular
        @Procedure("usp_obtenerRegistrosParaExportar")
        public List<Registro> obtenerRegistrosParaExportar (int codigoUsuario);

        //Lista los registros que reflejan ingresos de un usuarios en particular
        public List<Registro> findByImpactoGreaterThan0 (int codigoUsuario);

        //Lista los registros que reflejan egresos de un usuarios en particular
        public List<Registro> findByImpactoLessThan0 (int codigoUsuario);

        @Procedure("usp_obtenerSumatoriaGeneralImpactosYFecha")
        //Obtiene los impactos (ingresos y egresos) y sus fechas de un Usuario
        public List<List<Registro>> obtenerImpactosYFecha (int codigoUsuario);

        //Elimina un registro en particular
        @Procedure("usp_eliminarRegistro")
        public void eliminar (String codigoRegistro, int codigoUsuario);

}
```

Repositories: Registro

```java
public interface IUsuarioRepository extends JpaRepository<Usuario, Long> {

        //Busqueda de un usuario especifico mediante email
        public Usuario findByEmail (String email);

        //Eliminacion de un usuario especifico mediante codigo
        public void deleteById(int cod);

        //Actualizacion de un usuario especifico mediante nuevo objeto de tipo UsuarioDTO
        @Procedure("usp_actualizarUsuario")
        public int actualizar (Usuario usuario);

        //Creacion de un nuevo usuario mediante nuevo objeto de tipo UsuarioDTO
        public Usuario agregar (Usuario usuario);



}
```

Repositories: Usuario