		$ 60.000				
	Proyecto Plataforma de Tutorías en Línea Tutor Shop					
	Propuesta comercial del proyecto.					
	Actividades					$ 15.576.000
Sprint 1	**Objetivo:** Conformar un equipo de trabajo y definir roles.	**hrs**	2,50	$	150.000	
	1. Conformar equipo de trabajo	0,50				
	2. Crear una organización en GitHub	0,20				
	3. Crear un repositorio ejemplo de la organización, donde cada miembro **evidencie un commit**	0,10				
	• Descargar **Git(bash)**: https://git-scm.com/download/win • Descargar **GitHub Desktop**: https://desktop.github.com/ • Crear un nuevo repositorio e ignorar la carpeta **NODE** • Descargar **Node.js** https://nodejs.org/en/download • Descargar **Postman**, permite simular cliente REST (Respuesta del front): https://www.postman.com/downloads/	0,50				
Sprint 2	**Objetivo: Crear el frontend para la aplicación.**			$	8.400.000	
	Diseño y desarrollo de interfaces de usuario intuitivas y atractivas.	140,00				
	front					
	1. Crear una SPA (single-page application con React), para lo cual se debe crear la aplicación con el comando **npm create-react-app <nombreAplicación>**; esto generará todos los paquetes necesarios para dicha creación.					
	2. Crear la carpeta de los componentes del proyecto > **components** **>feaures.** >studentFormSlice.js Estados iniciales del formulario de registro del estudiante, datos de contacto y reserva de tutorías. >userSlice: Estados iniciales del usuario y persistencia local del usuario en el navegador. **>login.** Vista que permite el ingreso del usuario a la aplicación mediante el login. **>logout.** Vista que permite salir al usuario de la aplicación mediante el logout. **>models.** >statusModel.jsx. Definiciones de los estados de una reservación. **>schemas.** Esquemas de validación de los campos del formulario del estudiante, contacto y reserva de tutorías. **>services.** Realizar las peticiones HTTP asíncronas al back mediante la función integrada fetch de JavaScript. **>shared.** Vistas generales de la página como son el Navar, Home y Footer. **>store.** Árbol de objetos que almacena los estados de la aplicación. **>student.** Contiene las vistas de la información de registro, contacto, servicios y reservas de tutorías del estudiante. **>tutor.** Contiene las vistas de la información de registro, selección de asignaturas, aceptar o rechazar tutorías asignadas del tutor.	104,00				
	3. Crear la carpeta styles, en la cual deberán alojar los estilos que deseen usar en todo el proyecto. > **App.css**	8,00				
	4. Crear el archivo App.js, se definen las rutas que se usa la aplicación. > **App.js**	2,00				
	5. Crear el archivo Index.js, punto de inicio de ejecución de la aplicación. > **Index.js**	2,00				
	6. Test front-end	24,00				
Sprint 3	**Objetivo: Crear las funcionalidades backend de la aplicación**					
	back	93,60		$	5.616.000	
	Diseño de la Base de datos Relacional					
	1.Instalar Node.js: es un entorno de JavaScript para crear aplicaciones escalables y asíncronas en el lado del servidor	0,25				
	2.Instalar la base de datos relacional PostgreSQL	1,00				
	3. Crear la ficha técnica de la aplicación bajo la carpeta raíz usando **npm init**, como resultado se crea el archivo **package.json**	0,25				
		0,10				
	4. Instalar: **express**: Intercomunicador entre el back y el front, El framework backend más popular para Node.js, que te permite crear aplicaciones web escalables y preparadas para la empresa. **pg pg-hstore**: Driver de postgreSQL **sequelize**: (Object-Relational Mapping) para Node.js mediante npm i express pg pg-hstore sequelize					
	5. Configurar el servidor	1,00				
	6. Configurar conexión a la base de datos PosgreSQL	1,00				

7. Instalar para autorización, generar token y encriptar el password del usuario
bcryptjs: es una biblioteca de JavaScript que proporciona funciones para el hashing seguro de contraseñas mediante el algoritmo bcrypt.
jsonwentoken: tokens están diseñados para ser utilizados en la autenticación y la autorización de aplicaciones web y servicios API.
body-parser: es un middleware de Node.js que se utiliza para procesar los cuerpos de las solicitudes entrantes en las aplicaciones web.

mediante npm i bcryptjs jsonwentoken body-parser

	Modelado de datos			
	8. Crear el modelo de la base de datos relacional motor PosgreSQL Se crearon los siguientes modelos para atender con los requerimientos solicitados: **People** > Users, Students, Tutors. **Reservations** > ReservationType, Students, Tutors **Tutors** > TutorSubjects, Subjects	24,00		
	9. Servicio para el CRUD en el modelo estudiante: 1) Es posible **agregar** nuevos estudiantes. 2) Es posible **agregar** un estudiante como tutor. 3) Es posible **listar** los estudiantes. 4) Es posible **modificar** los estudiantes. 5) Es posible **eliminar** estudiantes. Se crea el modelo de estudiante: studentModel.is Se crea el dto de estudiante: requestPeople.is Se crea el servicio (Controlador) de estudiante: studentController.js Se crea las rutas (endpoints) de estudiante: studentRoutes.js	8,00		
	10. Servicio para el CRUD en el modelo tutor: 1) Es posible **agregar** nuevos tutores. 2) Es posible agregar un tutor como estudiante. 3) Es posible **listar** los tutores. 4) Es posible **modificar** los tutores. 5) Es posible **eliminar** tutores. Se crea el modelo de tutor: tutorModel.js Se crea el dto de tutor: requestTutor.js Se crea el servicio (Controlador) de tutor: tutorController.js Se crea las rutas (endpoints) de tutor: tutorRoutes.js	8,00		
	11. Servicio para el CRUD en el modelo reservación: 1) Es posible **agregar** nuevas reservaciones realizadas por el estudiante. 2) Es posible **listar** las reservaciones realizadas por un estudiante. 3) Es posible **listar** las reservaciones asignadas a tutor. 3) Es posible **modificar** la reservación por su id. 4) Es posible **eliminar** la reservación por su id. Se crea el modelo de reservación: reservationModel.js Se crea el dto de tutor: requestReservation.js Se crea el servicio (Controlador) de tutor: reservationController.js Se crea las rutas (endpoints) de tutor: reservationRoutes.js	8,00		
	12. Servicio para el CRUD en el modelo asignaturas de un tutor: 1) Es posible **agregar** nueva asignatura a un tutor. 2) Es posible **listar** la asignatura asignada a un tutor. 3) Es posible **eliminar** la asignatura por su id. Se crea el modelo de reservación: tutorSubjectModel.js Se crea el dto de tutor: requestTutorSubject.js Se crea el servicio (Controlador) de tutorSubject: tutorSubjectController.js Se crea las rutas (endpoints) de tutorSubject: tutorSubjectRoutes.js	8,00		
	13. Test back-end	24,00		
	13. Agregar autenticación, el sistema está protegido por login	8,00		
	14. Desplegar la interfaz, Es posible acceder a la interfaz a través de internet	2,00		
Sprint 4	**Entregables**		23,50 $	1.410.000,00
	1. Código de la aplicación en versiones (web) • Aplicar buenas prácticas y principios SOLID en todo el proyecto • Aplicar organización básica del proyecto			
	2. Crear una propuesta de proyecto comercial que tenga estimación de tiempo y costos.	1,00		
	3. Manual de usuario.	4,00		
	4. Manual del sistema.	16,00		
	5. Diseño de interfaces graficas de usuario.	1,00		
	6. Diseño de base de datos Relacional y/o No relacional.	1,00		
	7. Sustentación del proyecto en vivo.	0,50		