



Manual Técnico

11/7/2025

DWF

Herramientas y tecnologías que se utilizaron

Spring Boot: es un framework de código abierto que da soporte para el desarrollo de aplicaciones y páginas webs basadas en Java. Se trata de uno de los entornos más populares y ayuda a los desarrolladores back-end a crear aplicaciones con un alto rendimiento empleando objetos de java sencillos.

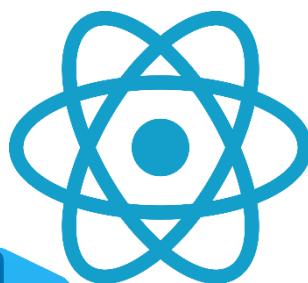
React native biblioteca de JavaScript de código abierto para crear interfaces de usuario interactivas.

Node.js es un entorno de ejecución de código JavaScript del lado del servidor, lo que permite a los desarrolladores usar JavaScript para crear aplicaciones web y herramientas de línea de comandos que no necesitan un navegador

XAMPP es un paquete de software gratuito y de código abierto que crea un servidor web local en tu computadora. El nombre es un acrónimo de Apache, MariaDB (antes MySQL), PHP y Perl, lo que permite a los desarrolladores crear y probar sitios y aplicaciones web en su máquina local antes de publicarlos en línea.

Visual studio code: es un editor de código fuente gratuito, ligero y potente para Windows, macOS y Linux. Es muy personalizable y extensible gracias a un amplio ecosistema de extensiones, lo que le permite soportar una gran variedad de lenguajes de programación y entornos de desarrollo. Está diseñado para ser eficiente e incluye funciones como resaltado de sintaxis, depuración, control de versiones con integración

de Git y autocompletado de código



Diagramas

Arquitectura

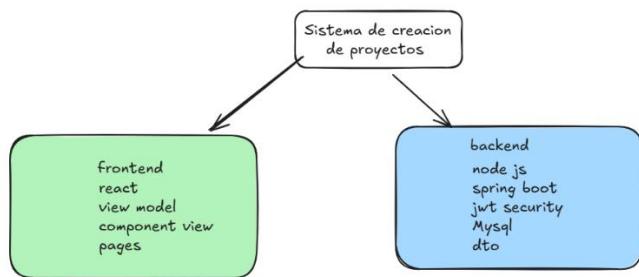


Diagrama de caso de uso

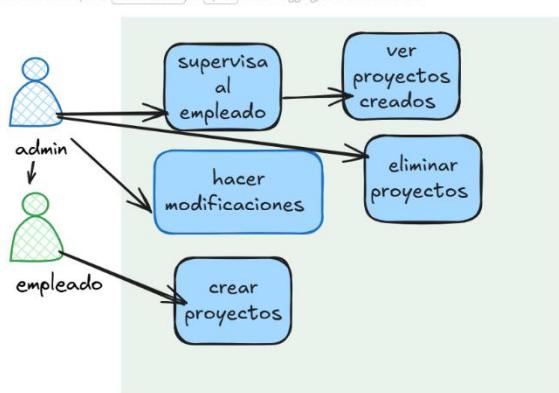
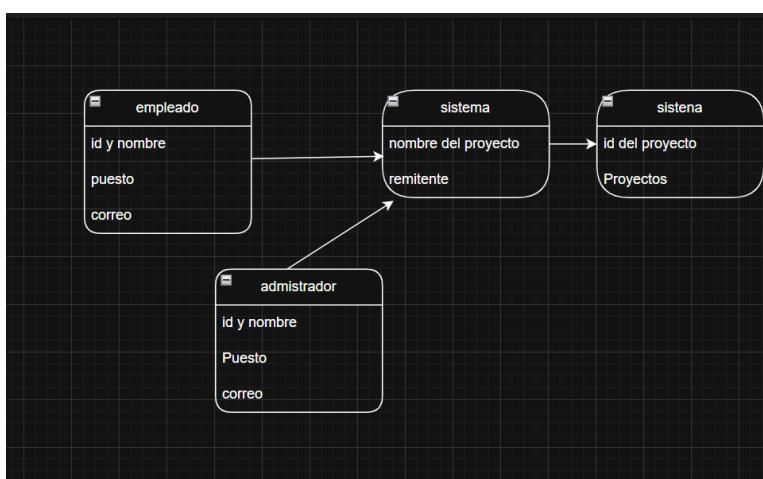
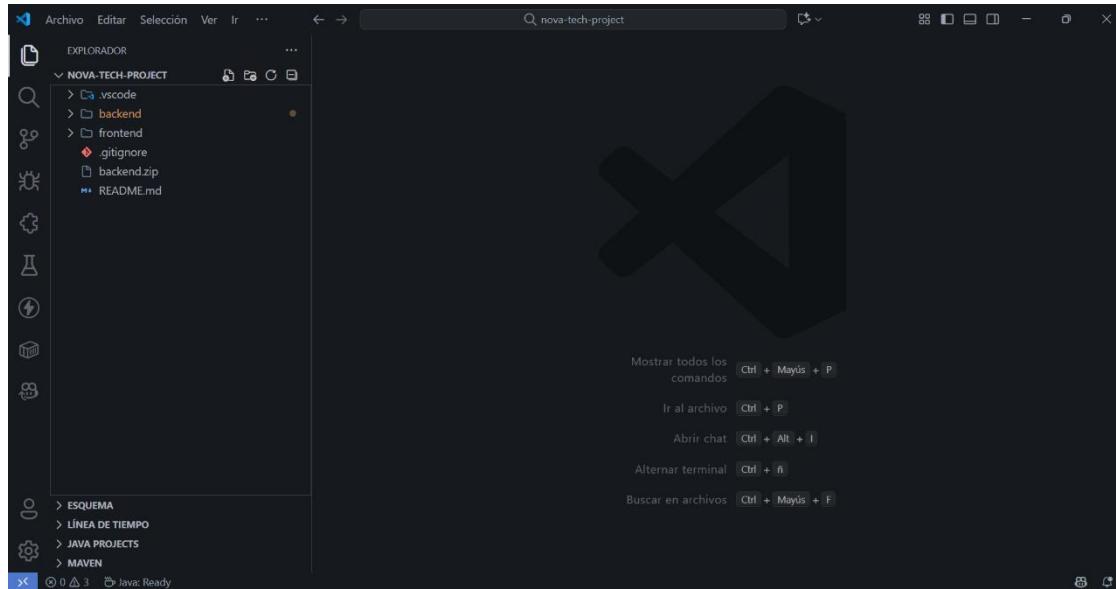


Diagrama de la base datos

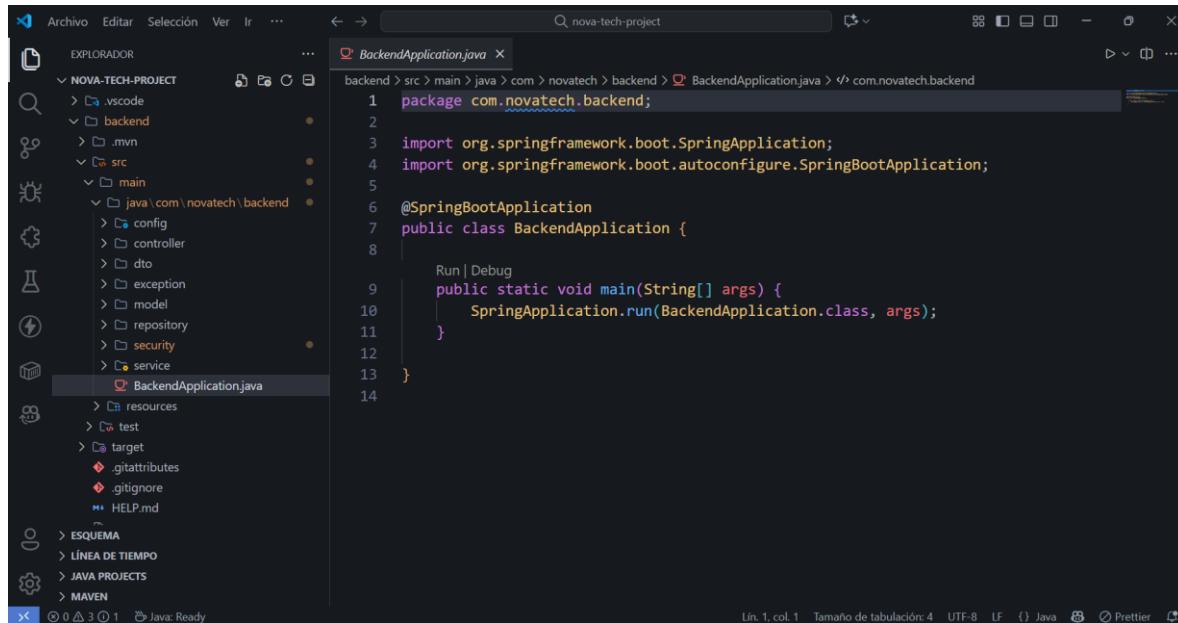


Estructura del proyecto

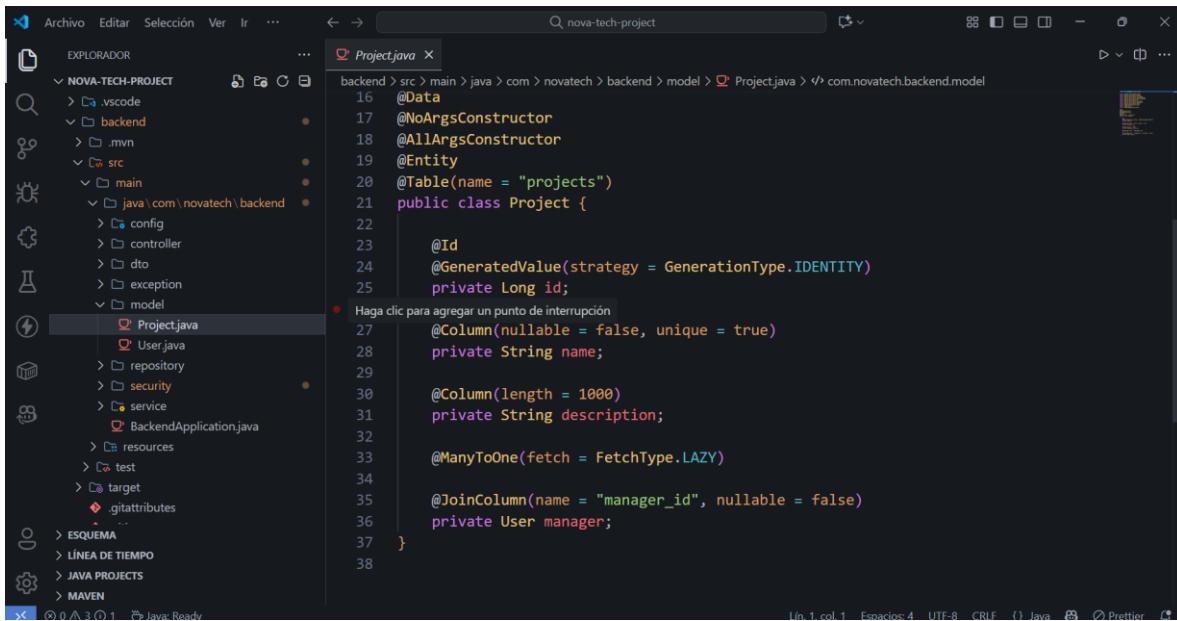
Esta separado por carpetas el front y backend



En la carpeta del backend esta toda la lógica del proyecto



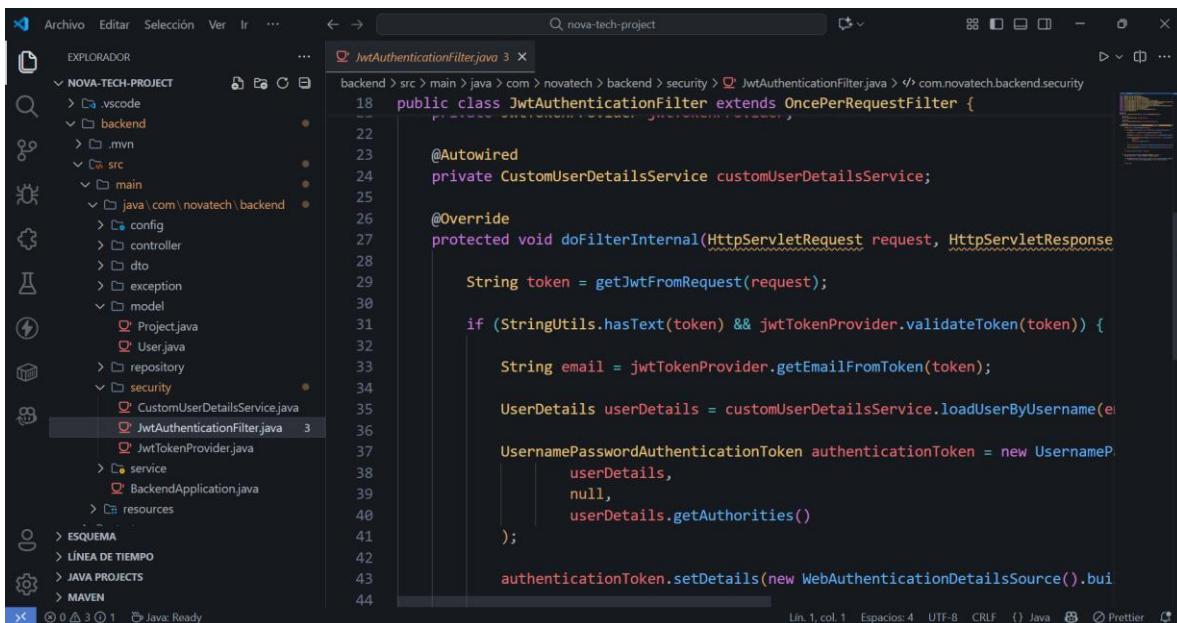
Observando algunas tenemos los modelos



The screenshot shows the VS Code interface with the file `Project.java` open in the editor. The code defines a `Project` entity with attributes `name` and `description`, and a relationship to a `User` manager. The code is annotated with `@Data`, `@Entity`, `@Table`, `@Id`, and `@GeneratedValue`.

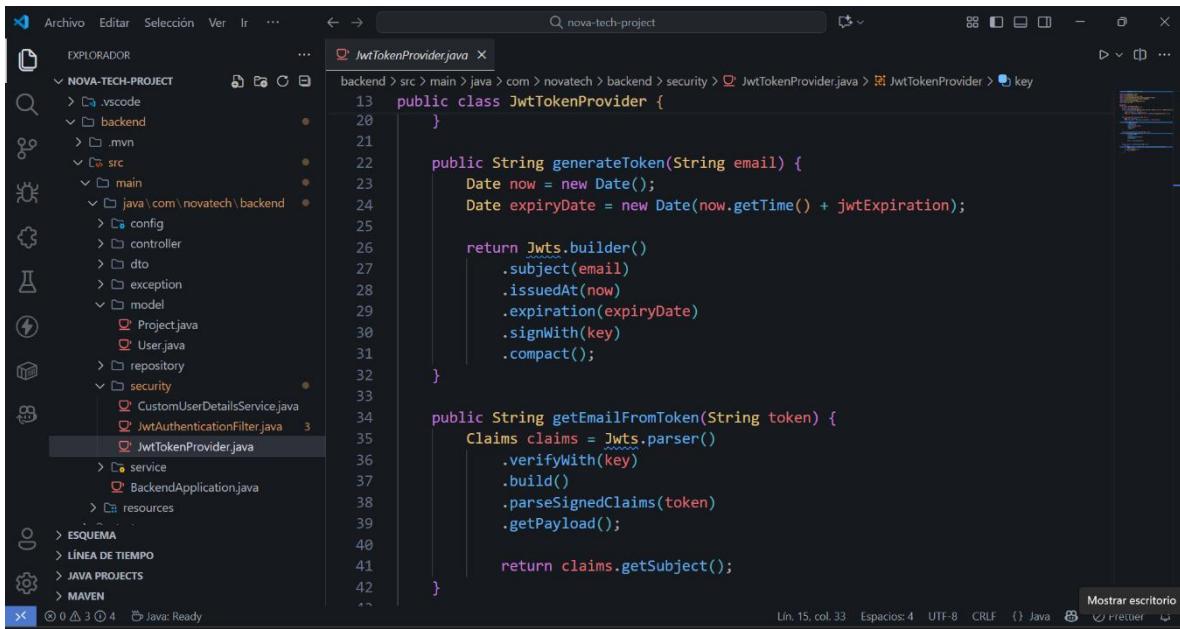
```
16  @Data
17  @NoArgsConstructor
18  @AllArgsConstructor
19  @Entity
20  @Table(name = "projects")
21  public class Project {
22
23      @Id
24      @GeneratedValue(strategy = GenerationType.IDENTITY)
25      private Long id;
26
27      @Column(nullable = false, unique = true)
28      private String name;
29
30      @Column(length = 1000)
31      private String description;
32
33      @ManyToOne(fetch = FetchType.LAZY)
34
35      @JoinColumn(name = "manager_id", nullable = false)
36      private User manager;
37  }
38
```

Tenemos también la carpeta de security es la seguridad de nuestro proyecto utilizamos jwt



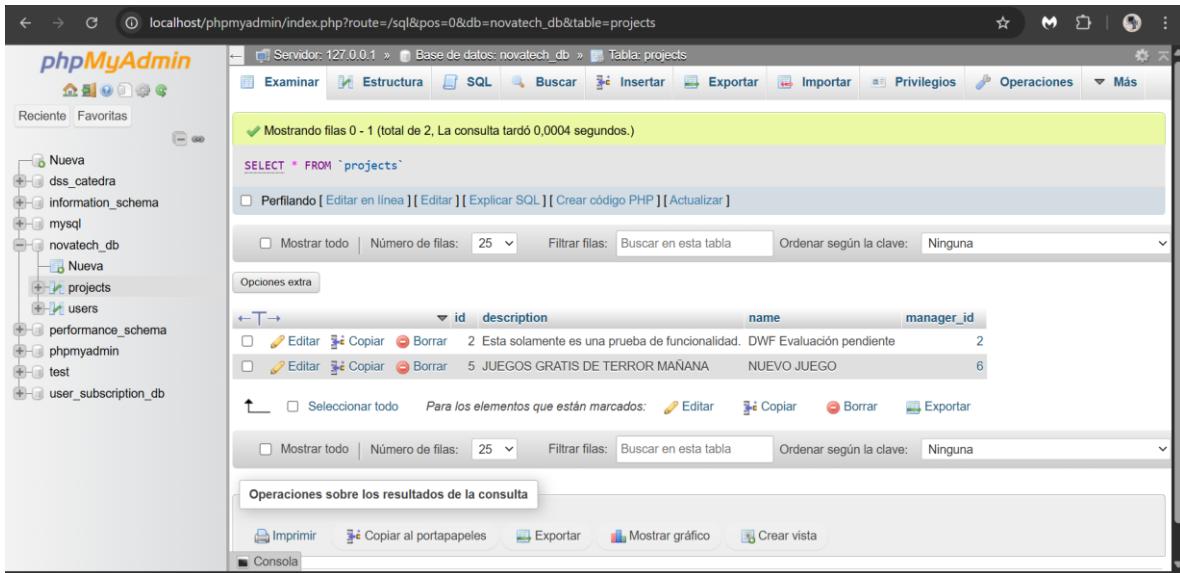
The screenshot shows the VS Code interface with the file `JwtAuthenticationFilter.java` open in the editor. The code implements a `OncePerRequestFilter` to handle JWT authentication. It uses `CustomUserDetailsService` to load users from a database based on the token.

```
18  public class JwtAuthenticationFilter extends OncePerRequestFilter {
19
20      @Autowired
21      private CustomUserDetailsService customUserDetailsService;
22
23      @Override
24      protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
25
26          String token = getJwtFromRequest(request);
27
28          if (StringUtils.hasText(token) && jwtTokenProvider.validateToken(token)) {
29
30              String email = jwtTokenProvider.getEmailFromToken(token);
31
32              UserDetails userDetails = customUserDetailsService.loadUserByUsername(email);
33
34              UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
35                  userDetails,
36                  null,
37                  userDetails.getAuthorities());
38
39              authenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
40
41          }
42
43      }
44
```



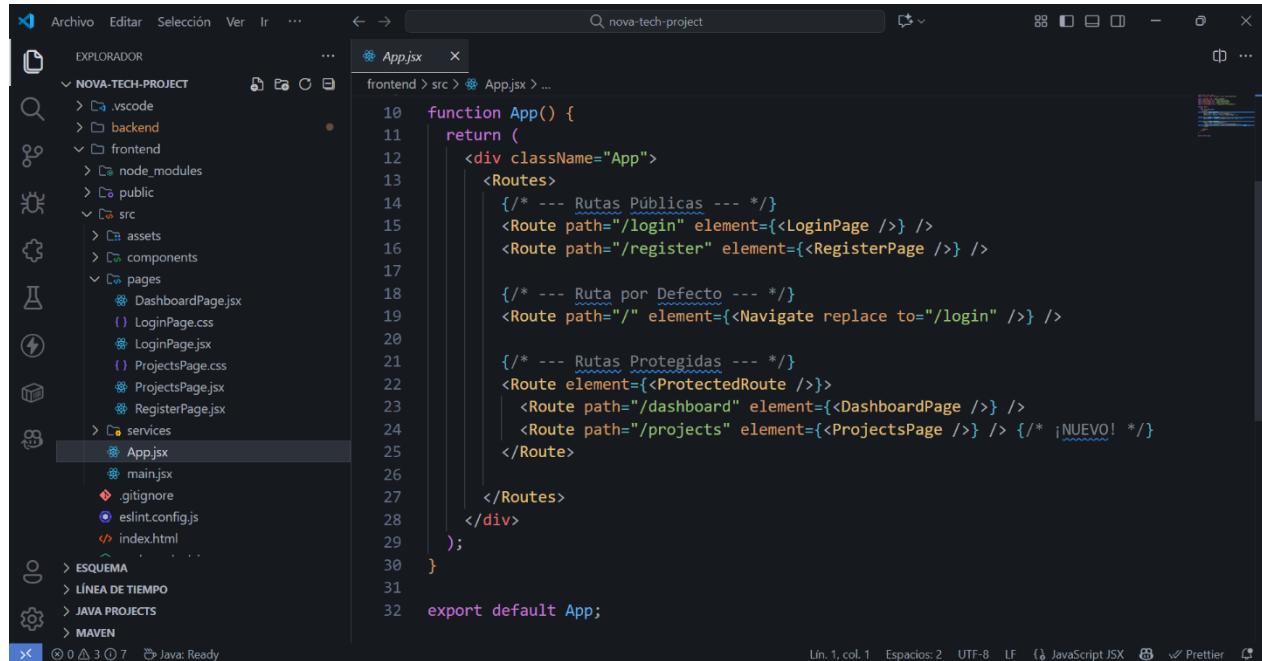
```
public class JwtTokenProvider {  
    public String generateToken(String email) {  
        Date now = new Date();  
        Date expiryDate = new Date(now.getTime() + jwtExpiration);  
  
        return Jwts.builder()  
            .subject(email)  
            .issuedAt(now)  
            .expiration(expiryDate)  
            .signWith(key)  
            .compact();  
    }  
  
    public String getEmailFromToken(String token) {  
        Claims claims = Jwts.parser()  
            .verifyWith(key)  
            .build()  
            .parseSignedClaims(token)  
            .getPayload();  
  
        return claims.getSubject();  
    }  
}
```

Tenemos nuestra base de datos que almacena los usuarios y proyectos



	id	description	name	manager_id
<input type="checkbox"/>	2	2 Esta solamente es una prueba de funcionalidad.	DWF Evaluación pendiente	2
<input type="checkbox"/>	5	5 JUEGOS GRATIS DE TERROR MAÑANA	NUEVO JUEGO	6

Y por parte del front tenemos lo siguiente aquí almacenamos todas nuestras vistas



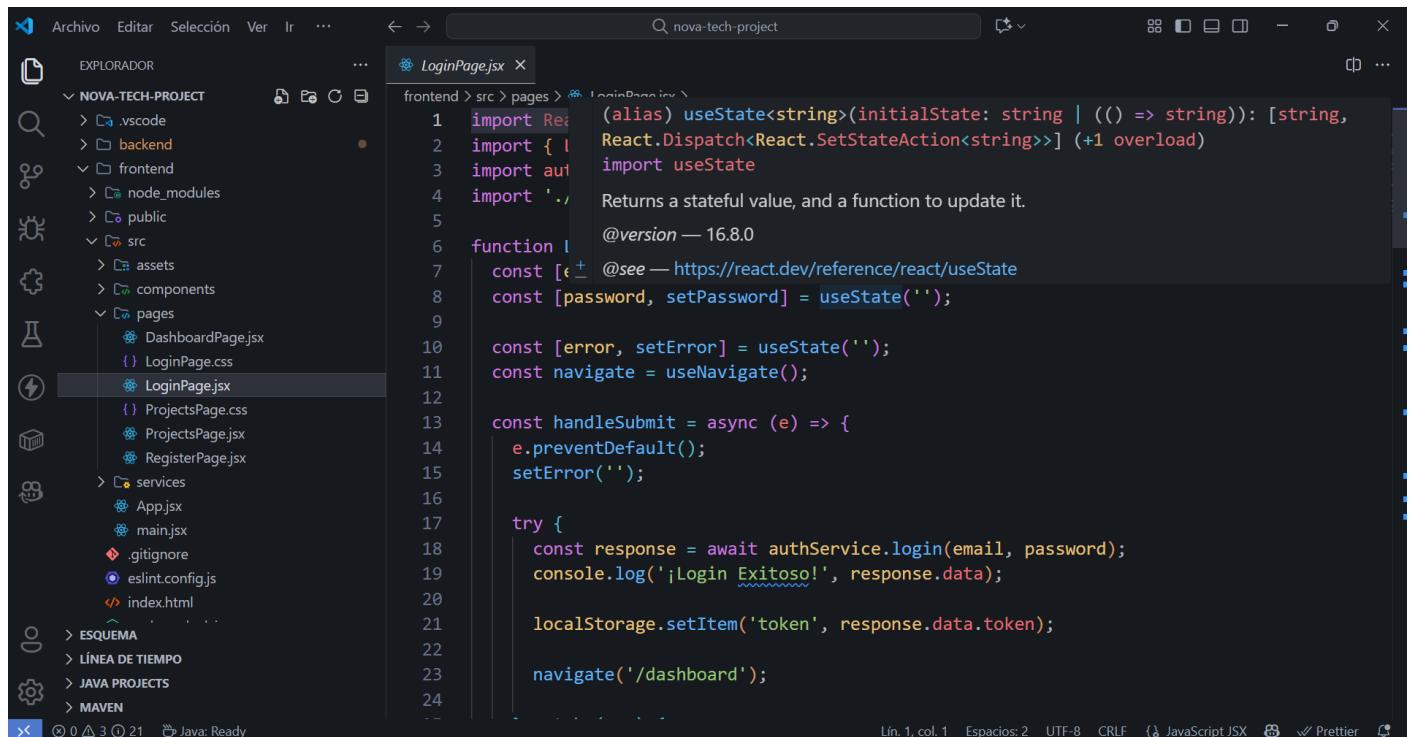
```
function App() {
  return (
    <div className="App">
      <Routes>
        {/* --- Rutas Públicas --- */}
        <Route path="/login" element={<LoginPage />} />
        <Route path="/register" element={<RegisterPage />} />

        {/* --- Ruta por Defecto --- */}
        <Route path="/" element={<Navigate replace to="/login" />} />

        {/* --- Rutas Protegidas --- */}
        <Route element={<ProtectedRoute />}>
          <Route path="/dashboard" element={<DashboardPage />} />
          <Route path="/projects" element={<ProjectsPage />} /> /* ¡NUEVO! */
        </Route>
      </Routes>
    </div>
  );
}

export default App;
```

Como la del login



```
import React, { useState } from 'react';
import authService from '../services/authService';
import { Navigate } from 'react-router-dom';

function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const [error, setError] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');

    try {
      const response = await authService.login(email, password);
      console.log('¡Login Exitoso!', response.data);

      localStorage.setItem('token', response.data.token);

      navigate('/dashboard');
    } catch (err) {
      setError(err.message);
    }
  };
}

export default LoginPage;
```