Documentación: Bcrypt

Uso en ProjectSENSEI — SenseiAPI

¿Qué es bcrypt?

bcrypt es una librería de hashing de contraseñas diseñada para ser lenta e irreversible. A diferencia del cifrado, el hash no puede descifrarse: solo puede compararse. Esto significa que aunque alguien acceda a la base de datos, no podrá obtener las contraseñas originales.

El factor de 'coste' controla cuántas veces se aplica el algoritmo. A mayor número, más seguro, pero más lento.

Instalación

```
npm install bcryptjs
```

Se usa bcryptjs en lugar de bcrypt, lo que lo hace compatible con cualquier entorno.

Fragmento 1 — Hashear contraseña al registrar usuario

Archivo: routes/auth.js

```
import bcrypt from 'bcryptjs'

router.post('/register', validate(registerSchema), async (req, res) => {
  const { username, password } = req.body
  const db = getDB()

  const existingUser = await db.collection('usuarios').findOne({ username })
  if (existingUser) {
    return res.status(400).json({ message: 'Usuario ya existe' })
  }

  const hashedPassword = await bcrypt.hash(password, 10)  // ← aquí

  await db.collection('usuarios').insertOne({
    username,
    password: hashedPassword,  // ← nunca se guarda la contraseña original
    role: 'user',
    createdAt: new Date()
  })
```

```
  res.status(201).json({ message: 'Usuario creado correctamente' })
})
```

¿Qué hace bcrypt.hash()?

La función ‘bcrypt.hash(password, saltRounds)’ toma la contraseña en texto plano y devuelve un hash seguro. El segundo argumento (10) es el número de 'salt rounds': cuántas iteraciones aplica el algoritmo internamente.

El resultado es una cadena como esta, que es lo que se guarda en MongoDB:

```
$2b$10$N9qo8uLOickgx2ZMRZoMyeIjZAgcfl7p92ldGxad68LJZdL17lhWy
```

Esta cadena contiene: el algoritmo ($2b$), el coste (10), el factor de coste y el hash, todo junto. bcrypt puede verificar una contraseña comparándola contra este string.

Fragmento 2 — Verificar contraseña al hacer login

Archivo: routes/auth.js

```
router.post('/login', validate(loginSchema), async (req, res) => {
  const { username, password } = req.body
  const db = getDB()

  const user = await db.collection('usuarios').findOne({ username })
  if (!user) return res.status(400).json({ message: 'Usuario no existe' })

  const validPassword = await bcrypt.compare(password, user.password)  // ← aquí
  if (!validPassword) return res.status(400).json({ message: 'Contraseña
incorrecta' })

  const token = jwt.sign(...)
  res.json({ token })
})
```

¿Qué hace bcrypt.compare()?

La función ‘bcrypt.compare(plainText, hash)’ compara la contraseña introducida por el usuario con el hash almacenado en la base de datos. Devuelve true si coinciden o false si no.

Nunca se desencripta el hash. bcrypt aplica el mismo algoritmo a la contraseña introducida y compara los resultados internamente. Esto es lo que lo hace seguro: incluso si alguien intercepta el proceso, no obtiene la contraseña original.

Fragmento 3 — Crear usuario admin al iniciar el servidor

Archivo: utils/createAdmin.js

```
import bcrypt from 'bcryptjs'
import { getDB } from '../config/db.js'

export const createAdminIfNotExists = async () => {
  const db = getDB()
  const username = process.env.ADMIN_USERNAME
  const password = process.env.ADMIN_PASSWORD

  const existingAdmin = await db.collection('usuarios').findOne({ username })
  if (existingAdmin) {
    console.log('Admin ya existe')
    return
  }

  const hashedPassword = await bcrypt.hash(password, 10)

  await db.collection('usuarios').insertOne({
    username,
    password: hashedPassword,
    role: 'admin',
    createdAt: new Date()
  })

  console.log(`Admin creado: usuario = ${username}`)
}
```

¿Por qué se hashea también aquí?

Aunque este usuario se crea automáticamente al arrancar el servidor (no viene de un formulario), la contraseña igualmente debe hashearse antes de guardarse. Las contraseñas nunca deben guardarse en texto plano en la base de datos, independientemente de su origen.

Las credenciales se leen desde variables de entorno (.env) para evitar que estén hardcodeadas en el código fuente.