Svelte 5 + Hono 🔥

Part 2: User & Auth backend

www.youtube.com/@GetterSethya

Base Repository

```
export interface AsyncBaseRepository<Model> {
      create(args: CreateArgs<Model>): Promise<Model>
      findById(args: FindByIdArgs): Promise<Model | null>
      list(args: ListArgs): Promise<ListReturn<Model>>
      update(args: UpdateArgs<Model>): Promise<Model>
      delete(args: DeleteArgs): Promise<boolean>
}
```

Base Repository

```ts
export type ListArgs = {
	sort: string
	order: OrderQuery
	q: string
	limit: number
	page: number
	[key: string]: any
}

export type CreateArgs<T> = {
	item: Omit<T, 'id' | 'created_at' | 'updated_at'>
}

export type FindByIdArgs = {
	id: number
	user: number
}

export type UpdateArgs<T> = {
	id: number
	user: number
	item: Partial<Omit<T, 'id'>>
}

export type DeleteArgs = {
	user: number
	id: number
}
```

User Repository

```ts
export class UserRepository implements Omit<AsyncBaseRepository<UserEntity>, 'create' | 'list'> {
    findById(args: Omit<FindByIdArgs, 'user'>): Promise<UserEntity | null> {
        throw new Error('Method not implemented.')
    }
    update(args: Omit<UpdateArgs<UserEntity>, 'user'>): Promise<UserEntity> {
        throw new Error('Method not implemented.')
    }
    delete(args: Omit<DeleteArgs, 'user'>): Promise<boolean> {
        throw new Error('Method not implemented.')
    }


}
```

User Controller

```
export const userController = new Hono()
    //
    .get('/id/:id', async (c) => {})
    .get('/current_user', async (c) => {})
```

Validator

```
npm install zod
```

```
npm i @hono/zod-validator
```

Custom validator

```ts
export function appValidator<
	Target extends keyof ValidationTargets,
	Output = any,
	Def extends ZodTypeDef = ZodTypeDef,
	Input = Output,
>(target: Target, schema: Zod.Schema<Output, Def, Input>) {
	return zValidator(target, schema, (result, c) => {
		const { data, ...rest } = result
		if (!result.success) {
			return appResponse(c, 'invalid input', 400, { ...rest })
		}
	})
}
```

Custom response

```
export function appResponse<T>(c: Context, message: string, status: ContentfulStatusCode, result: T) {
	return c.json({ message, result }, status)
}
```

Update user controller /:id use app validator

Setup JWT

```
npm install jose
```

```
export class JWT {
	private accessSecret: Uint8Array
	private refreshSecret: Uint8Array

	static alg = 'HS256'
	static issuer = 'sveltehono'
	static audience = 'urn:sveltehono:audience'
	static accessExp = '6h'
	static refreshExp = '7d'

	constructor(accessSecret: string, refreshSecret: string) {
		this.accessSecret = new TextEncoder().encode(accessSecret)
		this.refreshSecret = new TextEncoder().encode(refreshSecret)
	}

	public static decode(token: string) {}
	async createAccess(id: number) {}
	async createRefresh(id: number) {}
	async validateAccess(accessToken: string) {}
	async validateRefresh(refreshToken: string) {}
}
```

Auth controller

```
export const authController = new Hono()
     .post('/login', appValidator('form', loginValidator), async (c) => {})
     .post('/register', appValidator('form', registerValidator), async (c) => {})
     .post('/refresh', async (c) => {})
     .delete('/logout', async (c) => {})
```

```
npm install @node-rs/argon2
```

Auth repository

```
export class AuthenticationRepository implements Omit<AsyncBaseRepository<AuthEntity>,
'list'> {
     async login(args: LoginArgs) {}
     async register(args: RegisterArgs) {}
     async findByUser(args: FindByIdArgs) {}
     async findById(args: FindByIdArgs): Promise<AuthEntity | null> {}
     async update(args: UpdateArgs<AuthEntity>): Promise<AuthEntity> {}
     async delete(args: DeleteArgs): Promise<boolean> {}
}
```

JWT middleware

```
export const jwtMiddleware = async (c: Context<BlankEnv, '*', {}>, next: Next) => {
     const { JWT_ACCESS_SECRET, JWT_REFRESH_SECRET } = env<ENV>(c)
     const jwt = new JWT(JWT_ACCESS_SECRET, JWT_REFRESH_SECRET)
     const validAccess = await jwt.validateAccess(c.req.header('Authorization') || '')

     if (validAccess) {
          await next()
     } else {
          c.res = appResponse(c, 'unauthorize', 400, null)
     }
}
```

```
     .use('/current_user', jwtMiddleware)
     .use('/logout', jwtMiddleware)
```