Diagrama de clases del Proyecto

"SKIPUR"

Integrantes:

Chavez Oscullo Klever Enrique

Guacan Rivera Alexander David

Trejo Duque Alex Fernando

Fecha: 2025-07-15

Diagrama de clases SKIPUR

Los diagramas de clases del sistema de agendamiento de citas SKIPUR ofrecen una visión estructural de alto nivel de la arquitectura del software, reflejando la organización lógica y técnica del proyecto. Estos diagramas se han construido bajo los principios de la Arquitectura Limpia, lo que permite una separación clara de responsabilidades entre las distintas capas del sistema: dominio, aplicación e infraestructura.

A través de esta representación visual, es posible comprender cómo interactúan las distintas partes del sistema, cómo se encapsula la lógica de negocio en los casos de uso, y de qué manera las implementaciones concretas (como controladores, repositorios y servicios externos) se conectan a través de interfaces bien definidas. Esta estructura facilita el mantenimiento, fomenta la reutilización de componentes y mejora la capacidad de escalar o modificar el sistema sin afectar su núcleo funcional. En conjunto, los diagramas de clases sirven como una herramienta clave para documentar, analizar y comunicar la arquitectura técnica del proyecto.

Diagrama de arquitectura de capas

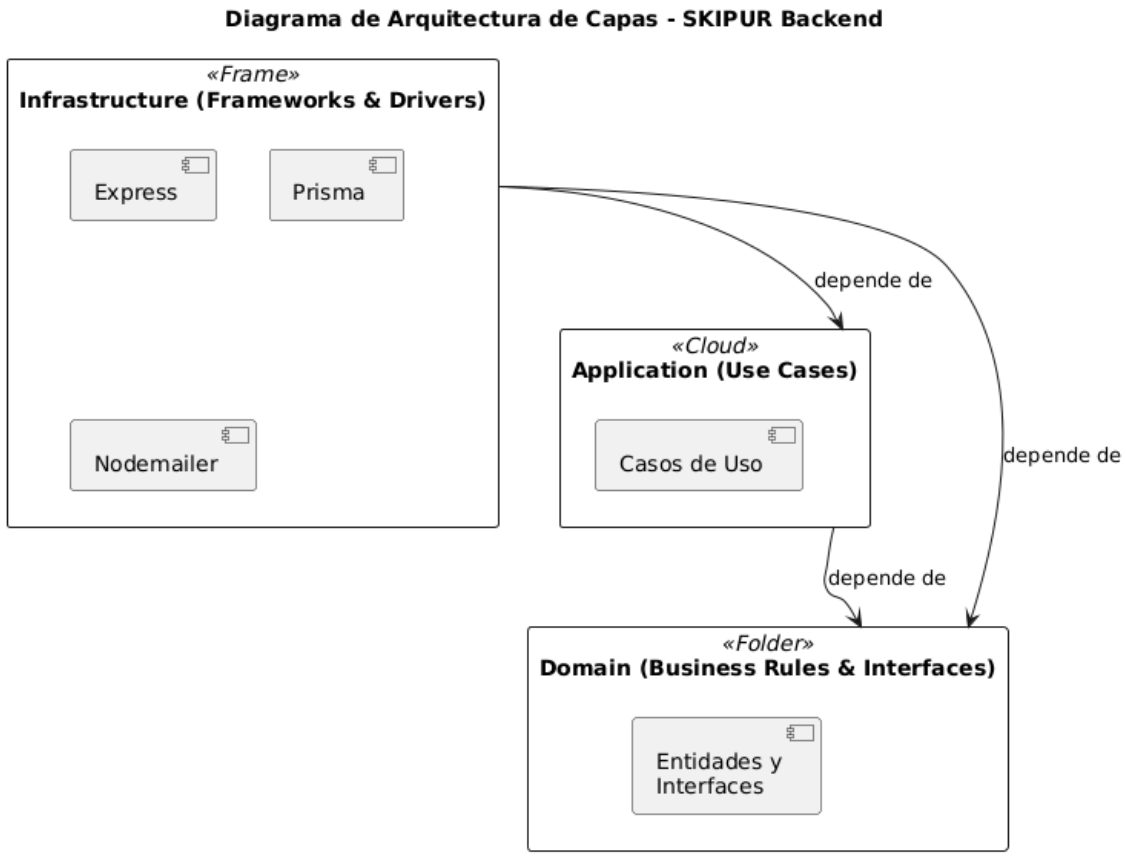
```
@startuml

title Diagrama de Arquitectura de Capas - SKIPUR Backend


skinparam packageStyle rectangle


package "Infrastructure (Frameworks & Drivers)" <<Frame>> as Infra {

   [Express]

   [Prisma]

   [Nodemailer]

}


package "Application (Use Cases)" <<Cloud>> as App {

   [Casos de Uso]

}


package "Domain (Business Rules & Interfaces)" <<Folder>> as Domain {

   [Entidades y\nInterfaces]

}


' Dependencias entre paquetes (capas)
```

Infra --> App : depende de

App --> Domain : depende de

Infra --> Domain : depende de

@enduml



Capa de dominio: Contratos y entidades

@startuml

title Capa de Dominio: Contratos y Entidades

skinparam packageStyle rectangle

skinparam classAttributeIconSize 0

package "Domain" {

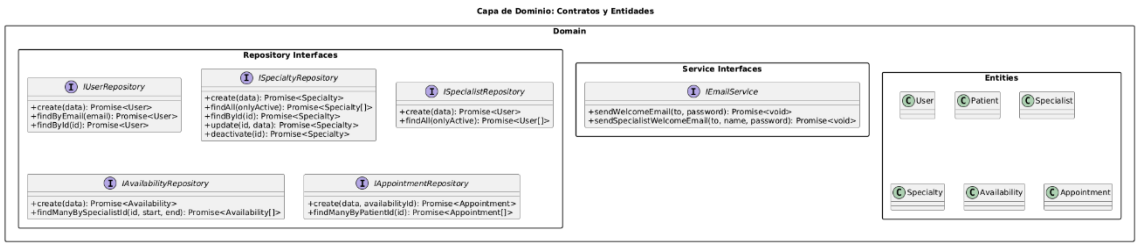
  ' Sección: Repository Interfaces

  package "Repository Interfaces" {

    interface IUserRepository {

```
        +create(data): Promise<User>
        +findByEmail(email): Promise<User>
        +findById(id): Promise<User>
    }

    interface ISpecialtyRepository {
        +create(data): Promise<Specialty>
        +findAll(onlyActive): Promise<Specialty[]>
        +findById(id): Promise<Specialty>
        +update(id, data): Promise<Specialty>
        +deactivate(id): Promise<Specialty>
    }

    interface ISpecialistRepository {
        +create(data): Promise<User>
        +findAll(onlyActive): Promise<User[]>
    }

    interface IAvailabilityRepository {
        +create(data): Promise<Availability>
        +findManyBySpecialistId(id, start, end): Promise<Availability[]>
    }

    interface IAppointmentRepository {
        +create(data, availabilityId): Promise<Appointment>
        +findManyByPatientId(id): Promise<Appointment[]>
    }
}

' Sección: Service Interfaces
package "Service Interfaces" {
    interface IEmailService {
```

```
        +sendWelcomeEmail(to, password): Promise<void>
        +sendSpecialistWelcomeEmail(to, name, password): Promise<void>
    }
  }

  ' Sección: Entities
  package "Entities" {
    class User
    class Patient
    class Specialist
    class Specialty
    class Availability
    class Appointment
  }
}
@enduml
```



Flujo de colaboración: Controladores, Casos de uso, Repositorio

```
@startuml
title Flujo de Colaboración: Controladores -> Casos de Uso -> Repositorios

skinparam packageStyle rectangle
skinparam classAttributeIconSize 0

package "Infrastructure" {

  package "HTTP (Express)" <<Network>> {
    class SpecialistController << (C,#ADD1B2) >>
```

```
        class AvailabilityController << (C,#ADD1B2) >>
        class AuthController << (C,#ADD1B2) >>
        class SpecialtyController << (C,#ADD1B2) >>
        class AppointmentController << (C,#ADD1B2) >>
    }

    package "Database (Prisma)" {
        class PrismaUserRepository << (C,#ADD1B2) >>
        class PrismaSpecialtyRepository << (C,#ADD1B2) >>
    }
}

package "Application (Use Cases)" {

    package "auth" {
        class LoginUserUseCase << (C,#ADD1B2) >>
        class RegisterUserUseCase << (C,#ADD1B2) >>
    }

    package "specialty" {
        class GetAllSpecialtiesUseCase << (C,#ADD1B2) >>
        class CreateSpecialtyUseCase << (C,#ADD1B2) >>
    }

    package "specialist" {
        class CreateSpecialistUseCase << (C,#ADD1B2) >>
    }

    package "availability" {
        class CreateAvailabilityUseCase << (C,#ADD1B2) >>
    }
```

```
    package "appointment" {
        class ReserveAppointmentUseCase << (C,#ADD1B2) >>
        class GetMyAppointmentsUseCase << (C,#ADD1B2) >>
    }
}

package "Domain (Interfaces)" {
    interface IAppointmentRepository << (I,#A9DCDF) >>
    interface IAvailabilityRepository << (I,#A9DCDF) >>
    interface IUserRepository << (I,#A9DCDF) >>
    interface ISpecialtyRepository << (I,#A9DCDF) >>
    interface IEmailService << (I,#A9DCDF) >>
}

' === Relaciones: Controladores -> Casos de Uso ===
AuthController --> LoginUserUseCase
AuthController --> RegisterUserUseCase

SpecialtyController --> GetAllSpecialtiesUseCase
SpecialtyController --> CreateSpecialtyUseCase

SpecialistController --> CreateSpecialistUseCase

AvailabilityController --> CreateAvailabilityUseCase

AppointmentController --> ReserveAppointmentUseCase
AppointmentController --> GetMyAppointmentsUseCase

' === Casos de Uso -> Interfaces del Dominio ===
LoginUserUseCase --> IUserRepository
RegisterUserUseCase --> IUserRepository
RegisterUserUseCase --> IEmailService
```

CreateSpecialtyUseCase --> ISpecialtyRepository

GetAllSpecialtiesUseCase --> ISpecialtyRepository

CreateSpecialistUseCase --> IUserRepository

CreateSpecialistUseCase --> IEmailService

CreateAvailabilityUseCase --> IAvailabilityRepository

ReserveAppointmentUseCase --> IAppointmentRepository

ReserveAppointmentUseCase --> IAvailabilityRepository

GetMyAppointmentsUseCase --> IAppointmentRepository

' === Implementaciones de interfaces en la capa de infraestructura ===

PrismaUserRepository ..|> IUserRepository

PrismaSpecialtyRepository ..|> ISpecialtyRepository

@enduml

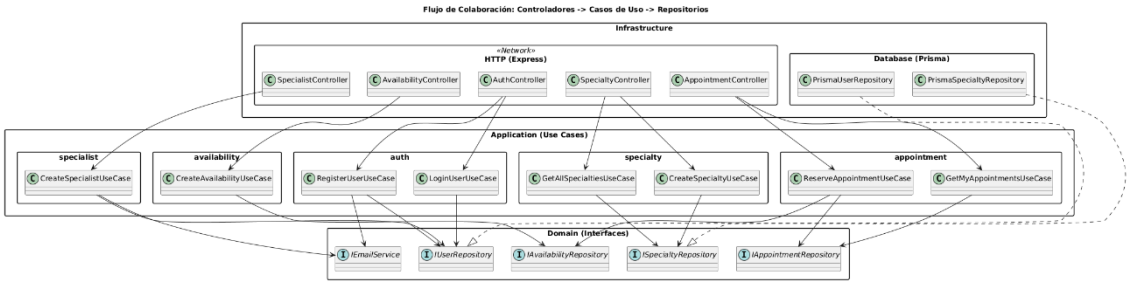


Diagrama de clases general

@startuml

title Diagrama de Clases General - Arquitectura SKIPUR

skinparam linetype ortho

skinparam packageStyle rectangle

```
skinparam classAttributeIconSize 0

package "Infrastructure (Frameworks & Drivers)" {
    package "HTTP (Express)" {
        class SpecialtyController <<C>>
        class AuthController <<C>>
        class authMiddleware <<C>>
    }

    package "Database (Prisma)" {
        class PrismaSpecialtyRepository <<C>>
        class PrismaUserRepository <<C>>
    }

    package "Services (Nodemailer)" {
        class NodemailerService <<C>>
    }
}

package "Application (Use Cases)" #LightGreen {
    package "specialty" {
        class CreateSpecialtyUseCase <<C>>
        class GetAllSpecialtiesUseCase <<C>>
    }

    package "auth" {
        class RegisterUserUseCase <<C>>
        class LoginUserUseCase <<C>>
    }

    package "specialist" {
        class CreateSpecialistUseCase <<C>>
```

```
    }

    package "availability" {
        class CreateAvailabilityUseCase <<C>>
    }

    package "appointment" {
        class ReserveAppointmentUseCase <<C>>
    }

    note right of appointment::ReserveAppointmentUseCase
        "La capa de Aplicación (Casos de Uso)\nusa las interfaces definidas en el Dominio."
    end note
}

package "Domain (Interfaces & Entities)" #AliceBlue {
    package "Repository Interfaces" {
        interface ISpecialtyRepository <<I>>
        interface IUserRepository <<I>>
        interface ISpecialistRepository <<I>>
        interface IAvailabilityRepository <<I>>
        interface IAppointmentRepository <<I>>
    }

    package "Service Interfaces" {
        interface IEmailService <<I>>
    }

    package "Entities" {
        class User
        class Appointment
        User "1" --> "*" Appointment : realiza/atiende
```

}

}

' Relaciones entre capas

SpecialtyController --> CreateSpecialtyUseCase

AuthController --> RegisterUserUseCase

authMiddleware --> LoginUserUseCase

CreateSpecialtyUseCase --> ISpecialtyRepository

GetAllSpecialtiesUseCase --> ISpecialtyRepository

CreateSpecialistUseCase --> ISpecialistRepository

CreateAvailabilityUseCase --> IAvailabilityRepository

ReserveAppointmentUseCase --> IAppointmentRepository

RegisterUserUseCase --> IUserRepository

LoginUserUseCase --> IUserRepository

PrismaSpecialtyRepository ..|> ISpecialtyRepository

PrismaUserRepository ..|> IUserRepository

NodemailerService ..|> IEmailService

RegisterUserUseCase --> IEmailService

@enduml