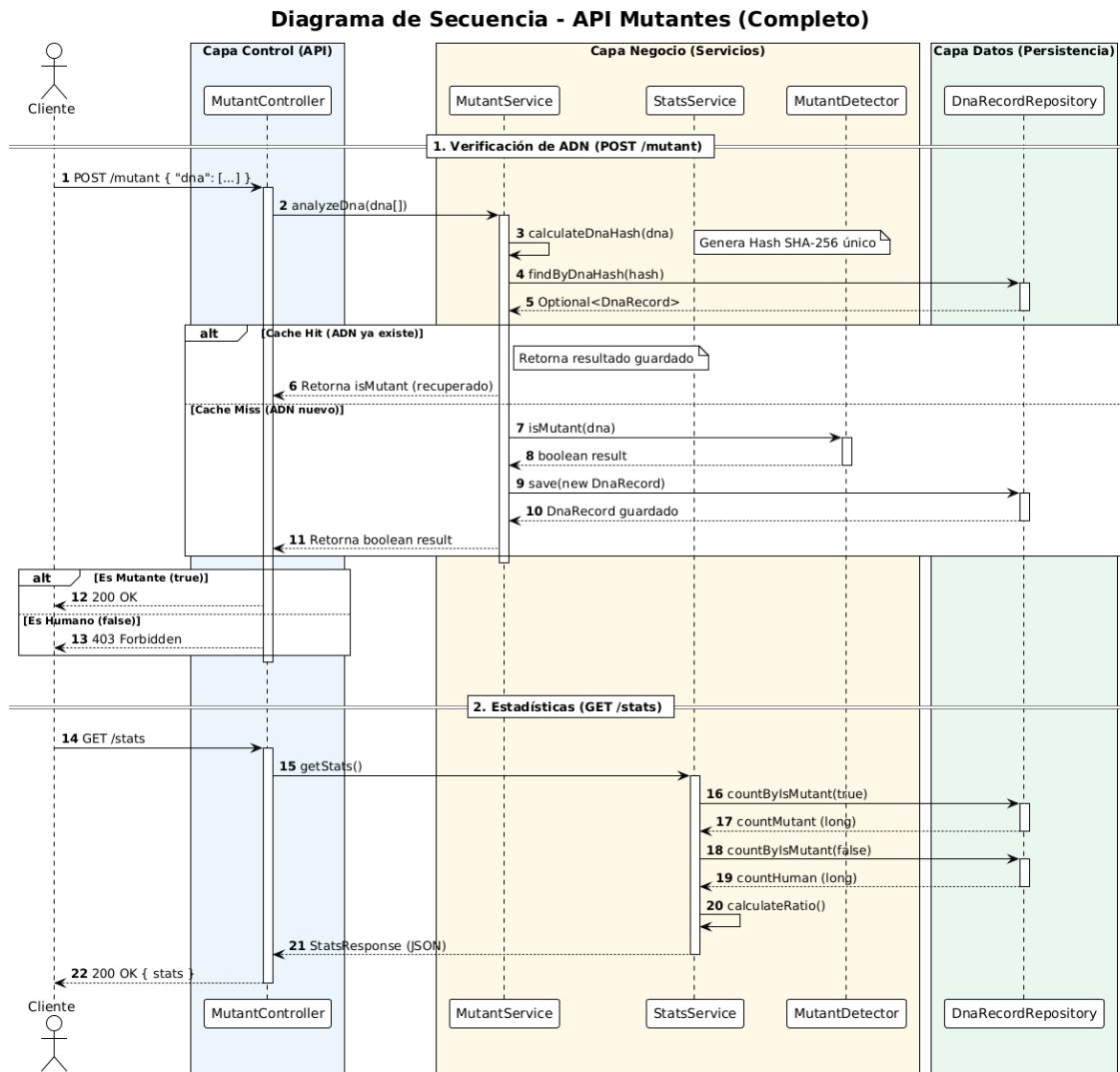


Diagrama de Secuencia



Código para pegar en PlantUML:

```

@startuml
!theme plain
autonumber

```

```

title Diagrama de Secuencia - API Mutantes (Completo)

actor "Cliente" as Client
participant "MutantController" as Controller

```

```

actor "MutantService" as Service
participant "StatsService" as Stats
participant "MutantDetector" as Detector
participant "DnaRecordRepository" as Repository

```

```

Client->>Controller: 1 POST /mutant {"dna": [...]}
Controller->>Service: 2 analyzedDna(dna[])
Service->>Controller: 3 calculateDnaHash(dna)
note over Controller: Genera Hash SHA-256 único
Controller->>Service: 4 findByDnaHash(hash)
Service->>Controller: 5 Optional<DnaRecord>
Controller->>Client: 6 Retorna isMutant (recuperado)
alt [Cache Hit (ADN ya existe)]
else [Cache Miss (ADN nuevo)]
    Service->>Controller: 7 isMutant(dna)
    Controller->>Client: 8 boolean result
    Controller->>Service: 9 save(new DnaRecord)
    Service->>Controller: 10 DnaRecord guardado
    Controller->>Client: 11 Retorna boolean result
end
alt [Es Mutante (true)]
Client->>Controller: 12 200 OK
else [Es Humano (false)]
Client->>Controller: 13 403 Forbidden
end
Client->>Controller: 14 GET /stats
Controller->>Service: 15 getStats()
Service->>Controller: 16 countByIsMutant(true)
Controller->>Service: 17 countMutant (long)
Controller->>Service: 18 countByIsMutant(false)
Controller->>Service: 19 countHuman (long)
Controller->>Service: 20 calculateRatio()
Controller->>Client: 21 StatsResponse (JSON)
Client->>Controller: 22 200 OK { stats }

```

```
participant "MutantService" as MutService
participant "StatsService" as StatService
participant "DnaRecordRepository" as Database
participant "MutantDetector" as Detector
```

```
box "Capa Control (API)" #EBF5FB
```

```
    participant Controller
end box
```

```
box "Capa Negocio (Servicios)" #FEF9E7
```

```
    participant MutService
    participant StatService
    participant Detector
end box
```

```
box "Capa Datos (Persistencia)" #E9F7EF
```

```
    participant Database
end box
```

```
== 1. Verificación de ADN (POST /mutant) ==
```

```
Client -> Controller: POST /mutant { "dna": [...] }
activate Controller
```

```
Controller -> MutService: analyzeDna(dna[])
activate MutService
```

```
MutService -> MutService: calculateDnaHash(dna)
```

note right: Genera Hash SHA-256 único

MutService -> Database: findByDnaHash(hash)

activate Database

Database --> MutService: Optional<DnaRecord>

deactivate Database

alt Cache Hit (ADN ya existe)

note right of MutService: Retorna resultado guardado

MutService --> Controller: Retorna isMutant (recuperado)

else Cache Miss (ADN nuevo)

MutService -> Detector: isMutant(dna)

activate Detector

Detector --> MutService: boolean result

deactivate Detector

MutService -> Database: save(new DnaRecord)

activate Database

Database --> MutService: DnaRecord guardado

deactivate Database

MutService --> Controller: Retorna boolean result

end

deactivate MutService

alt Es Mutante (true)

Controller --> Client: 200 OK

```
else Es Humano (false)

Controller --> Client: 403 Forbidden

end
```

deactivate Controller

|||

== 2. Estadísticas (GET /stats) ==

```
Client -> Controller: GET /stats

activate Controller
```

```
Controller -> StatService: getStats()

activate StatService
```

```
StatService -> Database: countByIsMutant(true)

activate Database

Database --> StatService: countMutant (long)

deactivate Database
```

```
StatService -> Database: countByIsMutant(false)

activate Database

Database --> StatService: countHuman (long)

deactivate Database
```

```
StatService -> StatService: calculateRatio()
```

StatService --> Controller: StatsResponse (JSON)

deactivate StatService

Controller --> Client: 200 OK{ stats }

deactivate Controller

@enduml