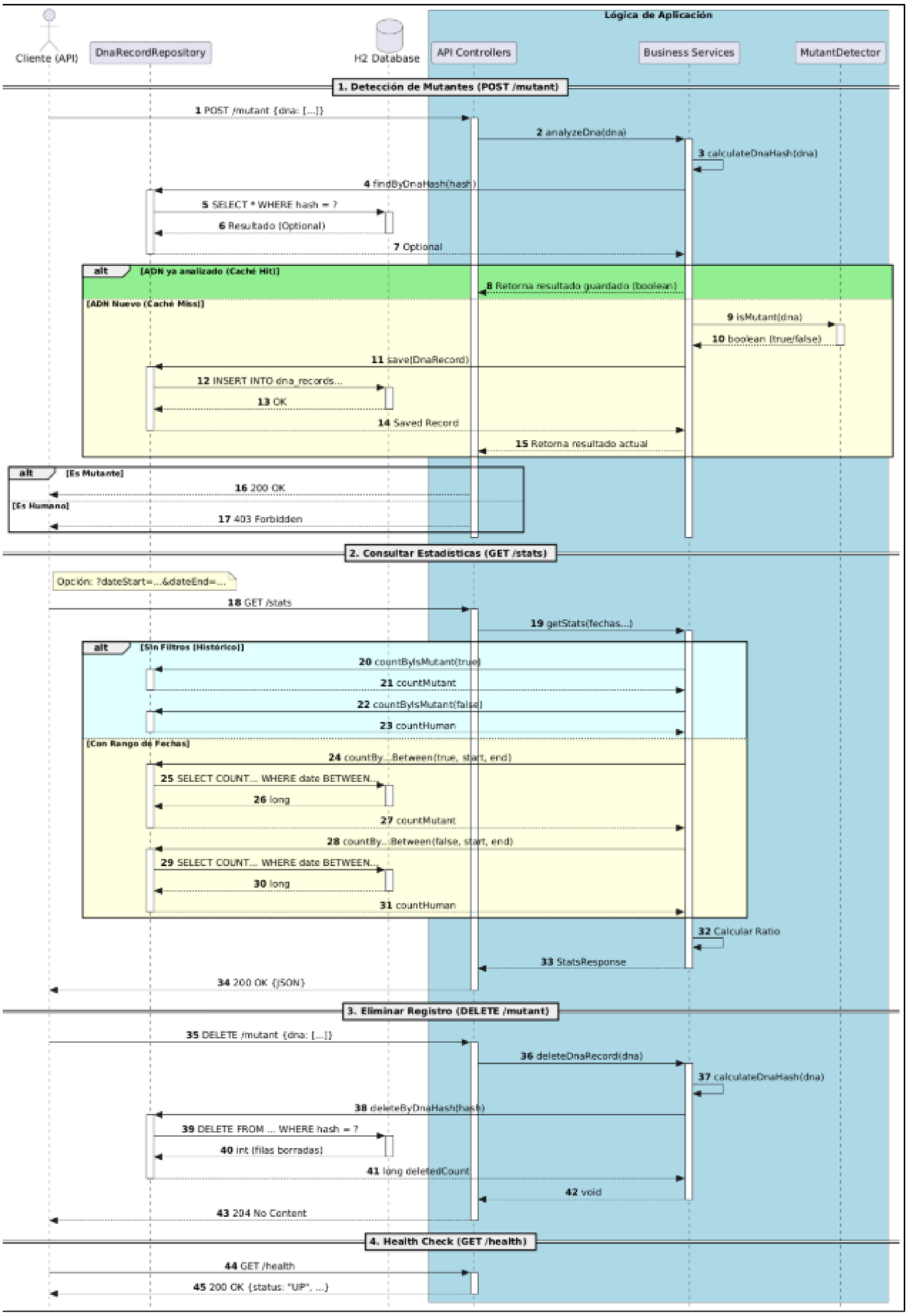
Este archivo contiene el diagrama de secuencia de todo el proyecto, se presenta en formato imagen y formato de código para diferentes visualizaciones:

Imagen del diagrama:



Código en PlantUML:

```
@startuml

autonumber

skinparam style strictuml

skinparam sequenceMessageAlign center


actor Cliente as "Cliente (API)"

participant Controller as "API Controllers"

participant Service as "Business Services"

participant Detector as "MutantDetector"

participant Repo as "DnaRecordRepository"

database DB as "H2 Database"


box "Lógica de Aplicación" #LightBlue

 participant Controller

 participant Service

 participant Detector

end box


== 1. Detección de Mutantes (POST /mutant) ==

Cliente -> Controller: POST /mutant {dna: [...]}

activate Controller

Controller -> Service: analyzeDna(dna)

activate Service

Service -> Service: calculateDnaHash(dna)

Service -> Repo: findByDnaHash(hash)

activate Repo

Repo -> DB: SELECT * WHERE hash = ?
```

```
activate DB
DB --> Repo: Resultado (Optional)
deactivate DB
Repo --> Service: Optional
deactivate Repo

alt #LightGreen ADN ya analizado (Caché Hit)
 Service --> Controller: Retorna resultado guardado (boolean)
else #LightYellow ADN Nuevo (Caché Miss)
 Service -> Detector: isMutant(dna)
 activate Detector
 Detector --> Service: boolean (true/false)
 deactivate Detector

 Service -> Repo: save(DnaRecord)
 activate Repo
 Repo -> DB: INSERT INTO dna_records...
 activate DB
 DB --> Repo: OK
 deactivate DB
 Repo --> Service: Saved Record
 deactivate Repo

 Service --> Controller: Retorna resultado actual
end

alt Es Mutante
 Controller --> Cliente: 200 OK
```

```
else Es Humano
 Controller --> Cliente: 403 Forbidden
end
deactivate Service
deactivate Controller

== 2. Consultar Estadísticas (GET /stats) ==
note right of Cliente: Opción: ?dateStart=...&dateEnd=...

Cliente -> Controller: GET /stats
activate Controller
Controller -> Service: getStats(fechas...)
activate Service

alt #LightCyan Sin Filtros (Histórico)
   Service -> Repo: countByIsMutant(true)
   activate Repo
   Repo --> Service: countMutant
   deactivate Repo
   Service -> Repo: countByIsMutant(false)
   activate Repo
   Repo --> Service: countHuman
   deactivate Repo
else #LightYellow Con Rango de Fechas
   Service -> Repo: countBy...Between(true, start, end)
   activate Repo
   Repo -> DB: SELECT COUNT... WHERE date BETWEEN...
   activate DB
```

```
  DB --> Repo: long
  deactivate DB
  Repo --> Service: countMutant
  deactivate Repo

  Service -> Repo: countBy...Between(false, start, end)
  activate Repo
  Repo -> DB: SELECT COUNT... WHERE date BETWEEN...
  activate DB
  DB --> Repo: long
  deactivate DB
  Repo --> Service: countHuman
  deactivate Repo
end

Service -> Service: Calcular Ratio
Service --> Controller: StatsResponse
deactivate Service
Controller --> Cliente: 200 OK {JSON}
deactivate Controller

== 3. Eliminar Registro (DELETE /mutant) ==
Cliente -> Controller: DELETE /mutant {dna: [...]}
activate Controller
Controller -> Service: deleteDnaRecord(dna)
activate Service
Service -> Service: calculateDnaHash(dna)
Service -> Repo: deleteByDnaHash(hash)
```

```
activate Repo
Repo -> DB: DELETE FROM ... WHERE hash = ?
activate DB
DB --> Repo: int (filas borradas)
deactivate DB
Repo --> Service: long deletedCount
deactivate Repo

Service --> Controller: void
deactivate Service
Controller --> Cliente: 204 No Content
deactivate Controller

== 4. Health Check (GET /health) ==
Cliente -> Controller: GET /health
activate Controller
Controller --> Cliente: 200 OK {status: "UP", ...}
deactivate Controller

@enduml
```