Construcción de una línea de producto de aplicaciones que serán ofrecidas a través de un marketplace en forma de Software como Servicio (SaaS)

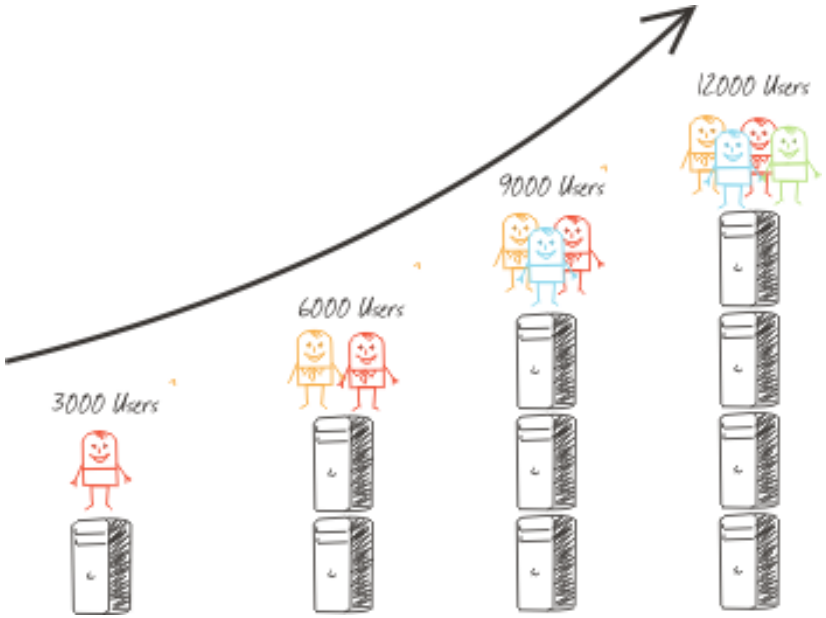
Universidad de los Andes

HEINSOHN MAPEO
MANEJO DE PROCESOS ORGANIZACIONALES

COLCIENCIAS
Departamento Administrativo de Ciencia, Tecnología e Innovación

Harold Castro, Ph.D.
hcastro@uniandes.edu.co
Director del Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
Bogotá, Colombia

Principal problema a resolver



¿Cómo desarrollar, mantener y comercializar soluciones de software para pequeñas y medianas empresas en Colombia sin que estas empresas tengan que realizar grandes inversiones en infraestructura de TI y que al mismo tiempo el proveedor pueda desarrollar modelos de negocio y ofrecer eficientemente soluciones que puedan llegar a tener alcance e impacto a nivel nacional, regional y/o mundial?

Simulador Plan de Pagos | + Crear | Generar Plan de Pagos | Cancel | Buscar por Cédula | Buscar por ID | Planes de Pagos

Tipo Documento
C.C.

Número Documento
(Ej. 1023546721)

Línea de Credito
(Ej. Automóvil)

Monto
(Ej. 200000000)

Perfil de Riesgo
Tipo 1

Plazo (Meses)
12

Días de Gracia
5

Fecha de Cuota #1
(Ej. 11/01/14)

Tasa (%)
(Ej. 0.08)

Valor A Girar
(Ej. 10000)

Definición de un prototipo SaaS

PROYECCIÓN CON TASA SUBSIDIADA

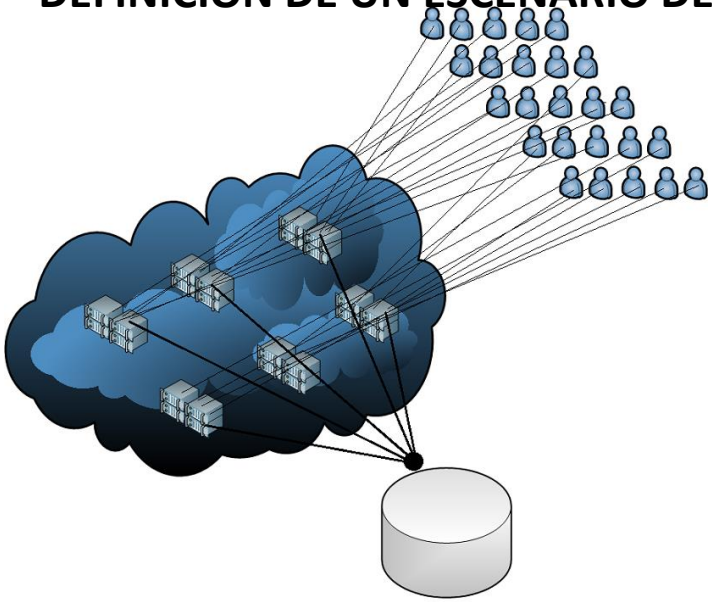
Cuotas	Valor Cuota	Saldo Restante
1	615.100	69.772.119
2	615.100	69.542.978
3	615.100	69.312.569
4	615.100	69.080.886
5	615.100	68.847.921
6	615.100	68.613.667
7	615.100	68.378.117
8	615.100	68.141.265
9	615.100	67.903.102

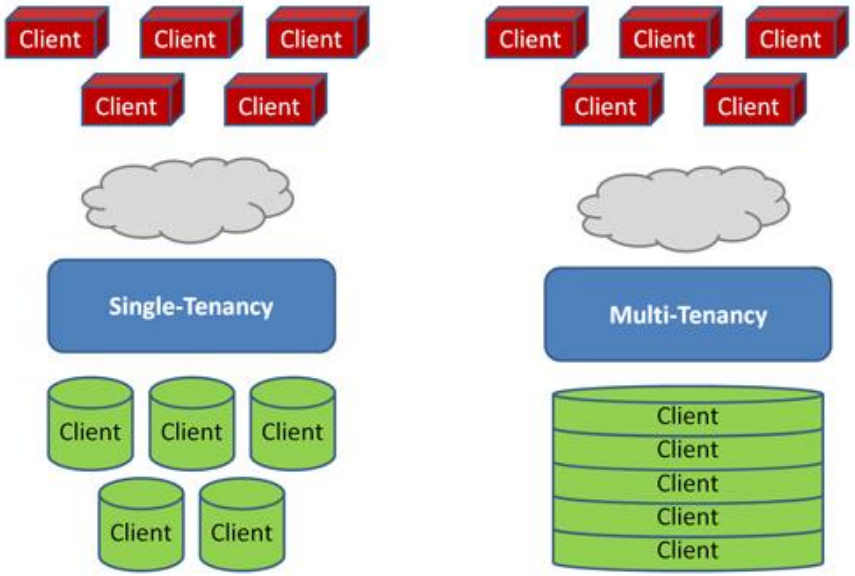
PROYECCIÓN CON TASA POLÍTICA

Cuotas	Valor Cuota	Saldo Restante
1	714.568	69.816.840
2	714.568	69.632.289
3	714.568	69.446.337
4	714.568	69.258.973
5	714.568	69.070.187
6	714.568	68.879.968
7	714.568	68.688.305
8	714.568	68.495.187
9	714.568	68.300.602

Para empezar a trabajar sobre cada uno de los frentes de trabajo se definió como prototipo de trabajo **un simulador de créditos bancarios**.

DEFINICIÓN DE UN ESCENARIO DE ESCALABILIDAD Y MULTITENANCY





- Cada empresa (tenant) crea una cuenta en la aplicación y configura unos parámetros básicos del simulador créditos.
- Cientos de usuarios internos (con usuario y password) de cada empresa (tenant) ingresan a la aplicación a realizar simulaciones.
- Un usuario puede consultar y guardar los diferentes planes de pago generados (tanto parámetros de entrada como parámetros de salida).
- La generación de un plan de pagos puede tomar un tiempo de procesamiento considerable, entre 5 y 30 segundos.

Las problemáticas abordadas

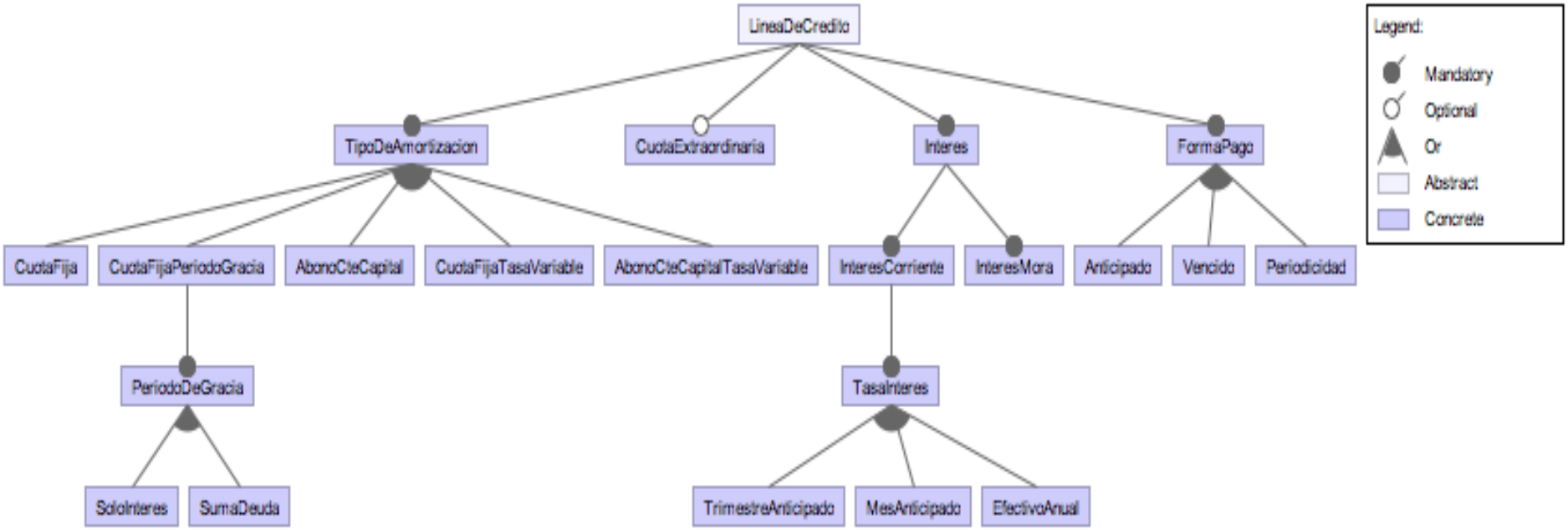
RETO 1

¿Cómo reducir los costos de desarrollo de aplicaciones individuales?

FRENTE DE TRABAJO 1

Entendimiento del dominio y desarrollo de una línea de productos bajo el modelo SaaS

Las problemáticas abordadas



Definición de una línea de producto de software (SPL) bajo el modelo SaaS

La problemática a solucionar

RETO 2

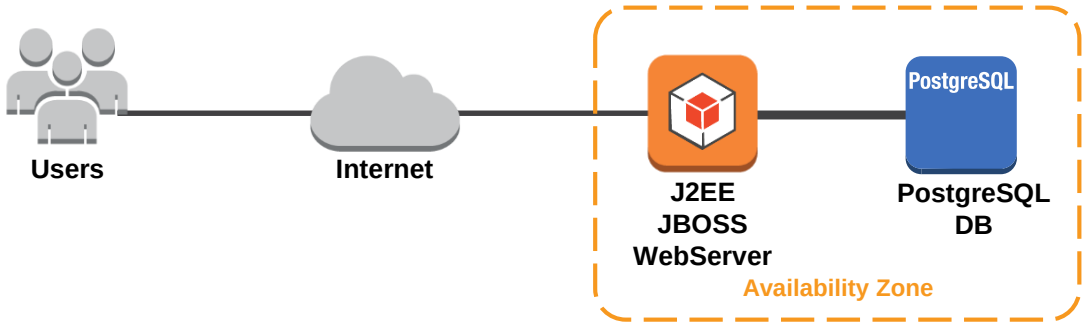
¿Cómo desarrollar aplicaciones escalables bajo el modelo SaaS (arquitecturas, tecnologías, etc.)?

FRENTE DE TRABAJO 2

Selección y evaluación de arquitecturas y tecnologías para construir aplicaciones escalables

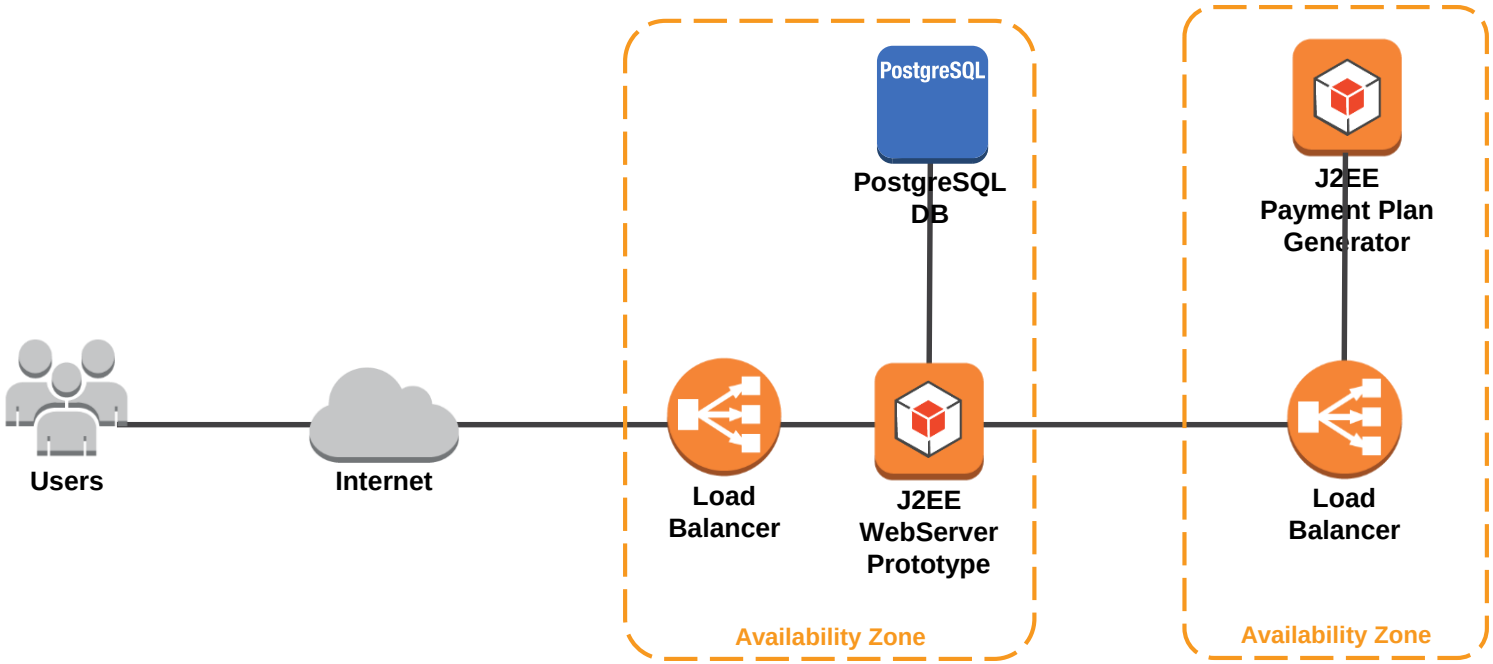
EVOLUCIÓN DE LA ARQUITECTURA 1

APLICACIÓN ACTUAL J2EE

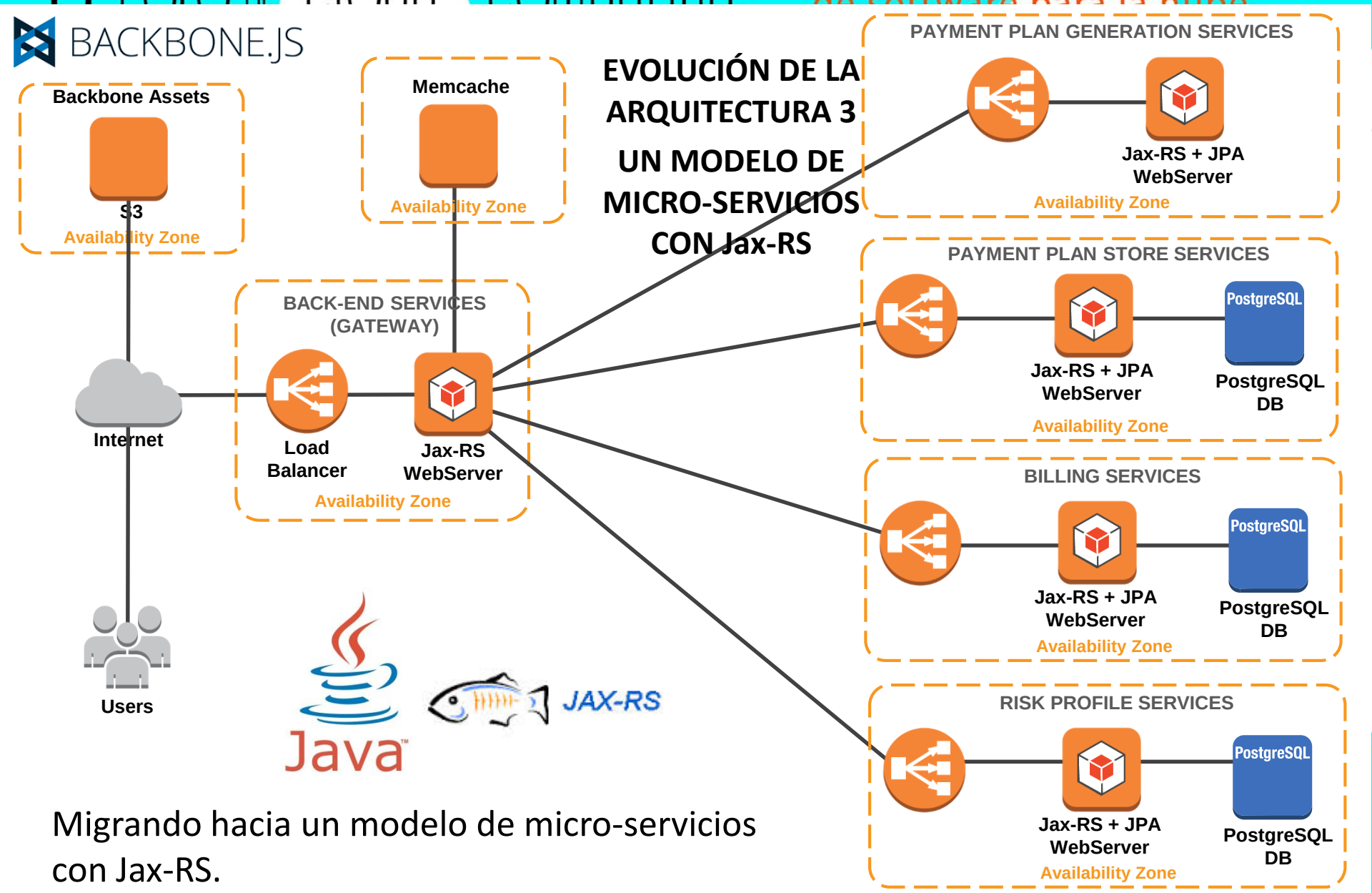


Migrar una aplicación monolítica tradicional a un modelo SaaS requiere de cambios a nivel de arquitectura.

EVOLUCIÓN DE LA ARQUITECTURA 2
DESACOPLANDO LA GENERACIÓN DE PLANES DE PAGO



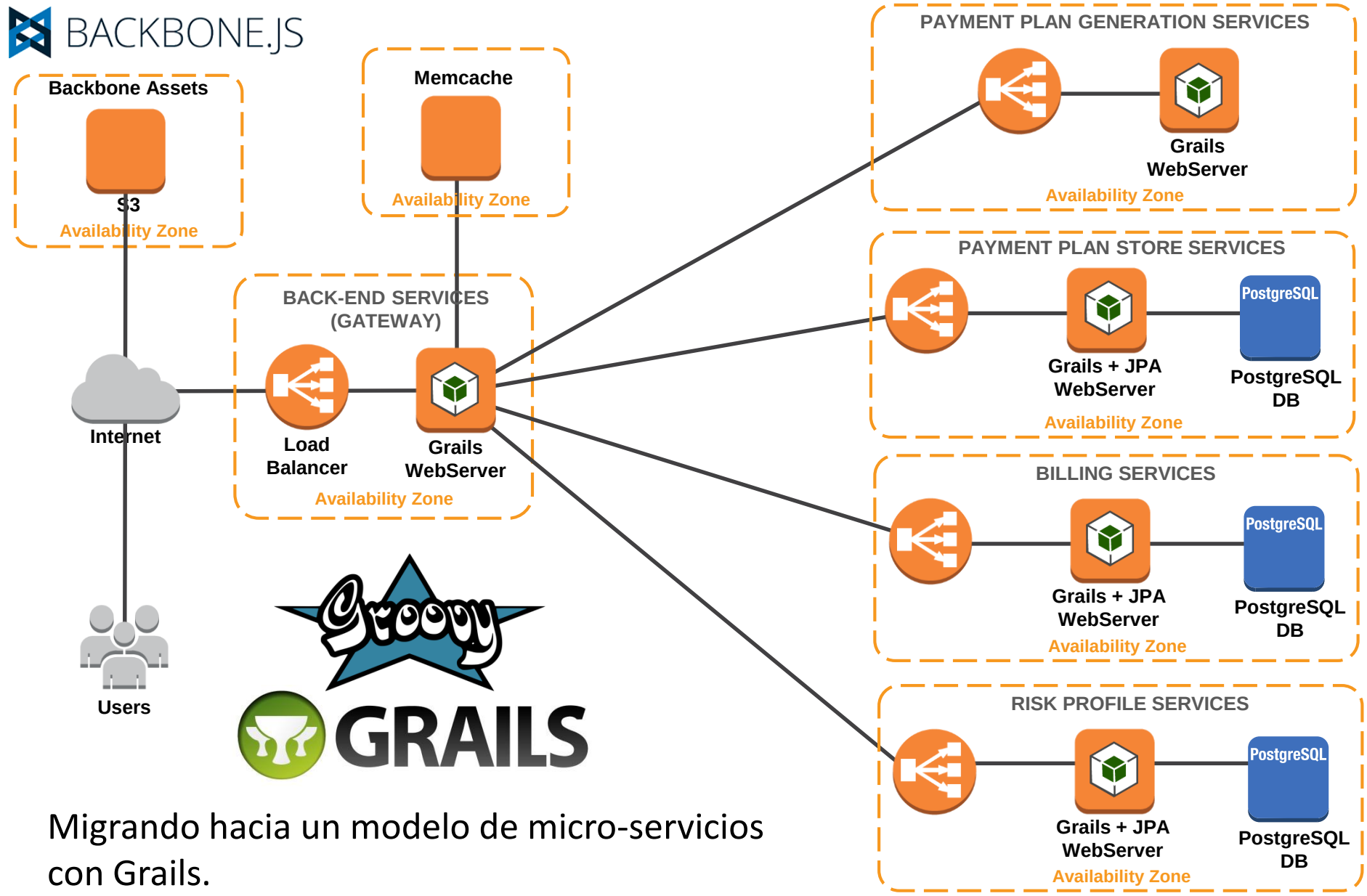
Desacoplando las tareas de intensivas en cómputo de las tareas con tiempo de respuesta cortos.



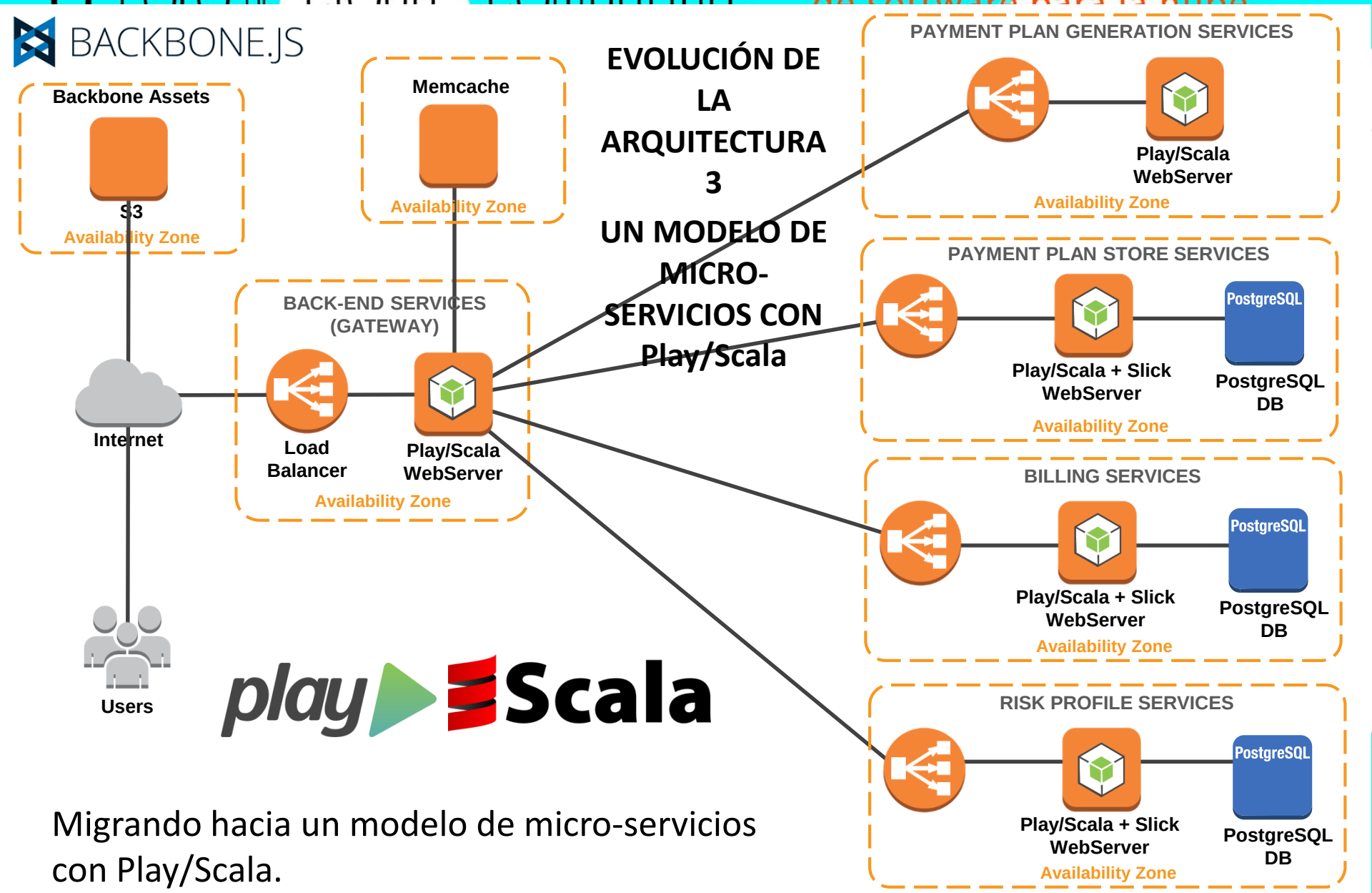
Migrando hacia un modelo de micro-servicios con Jax-RS.

EVOLUCIÓN DE LA ARQUITECTURA 3

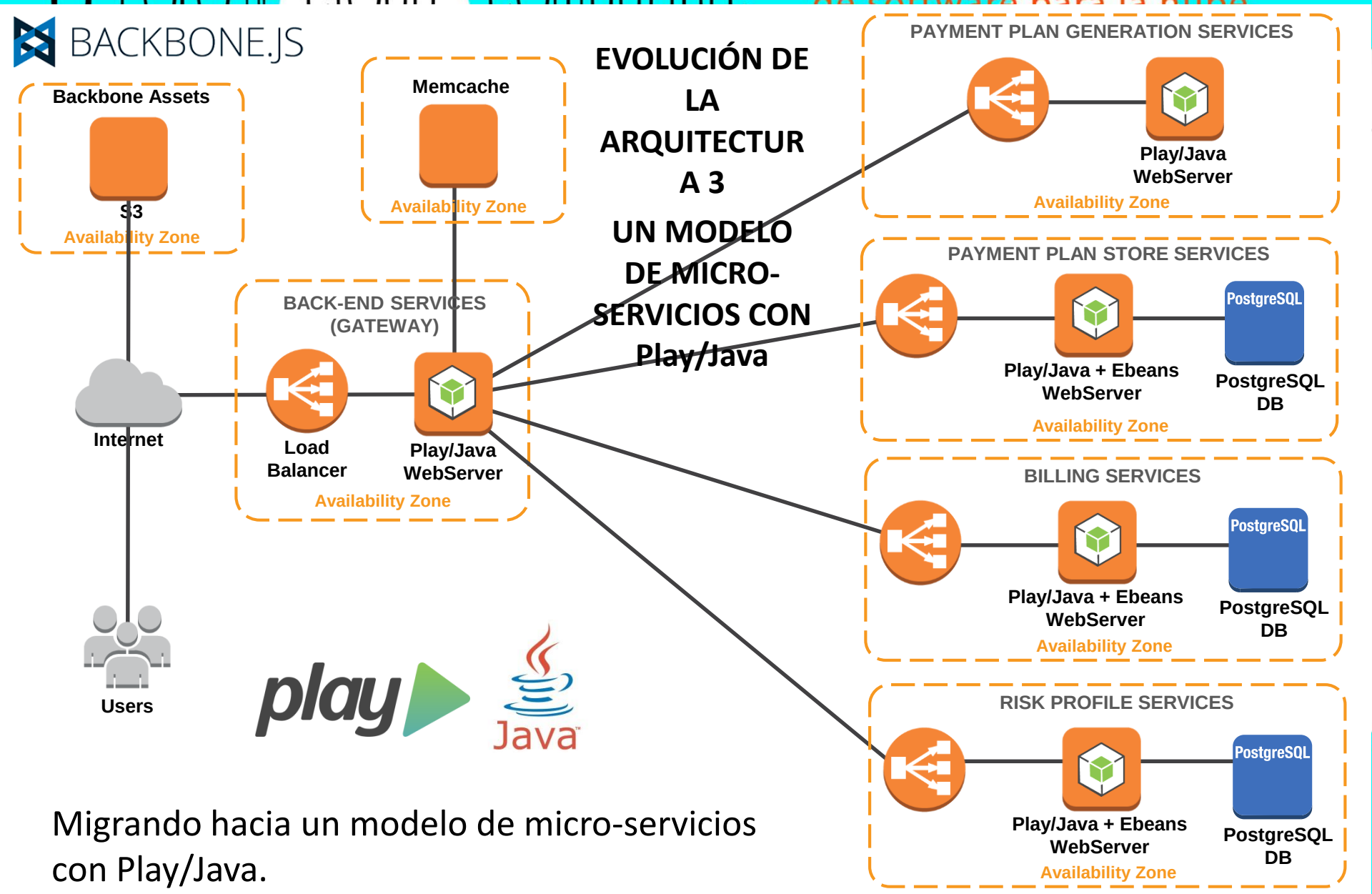
UN MODELO DE MICRO-SERVICIOS CON Grails



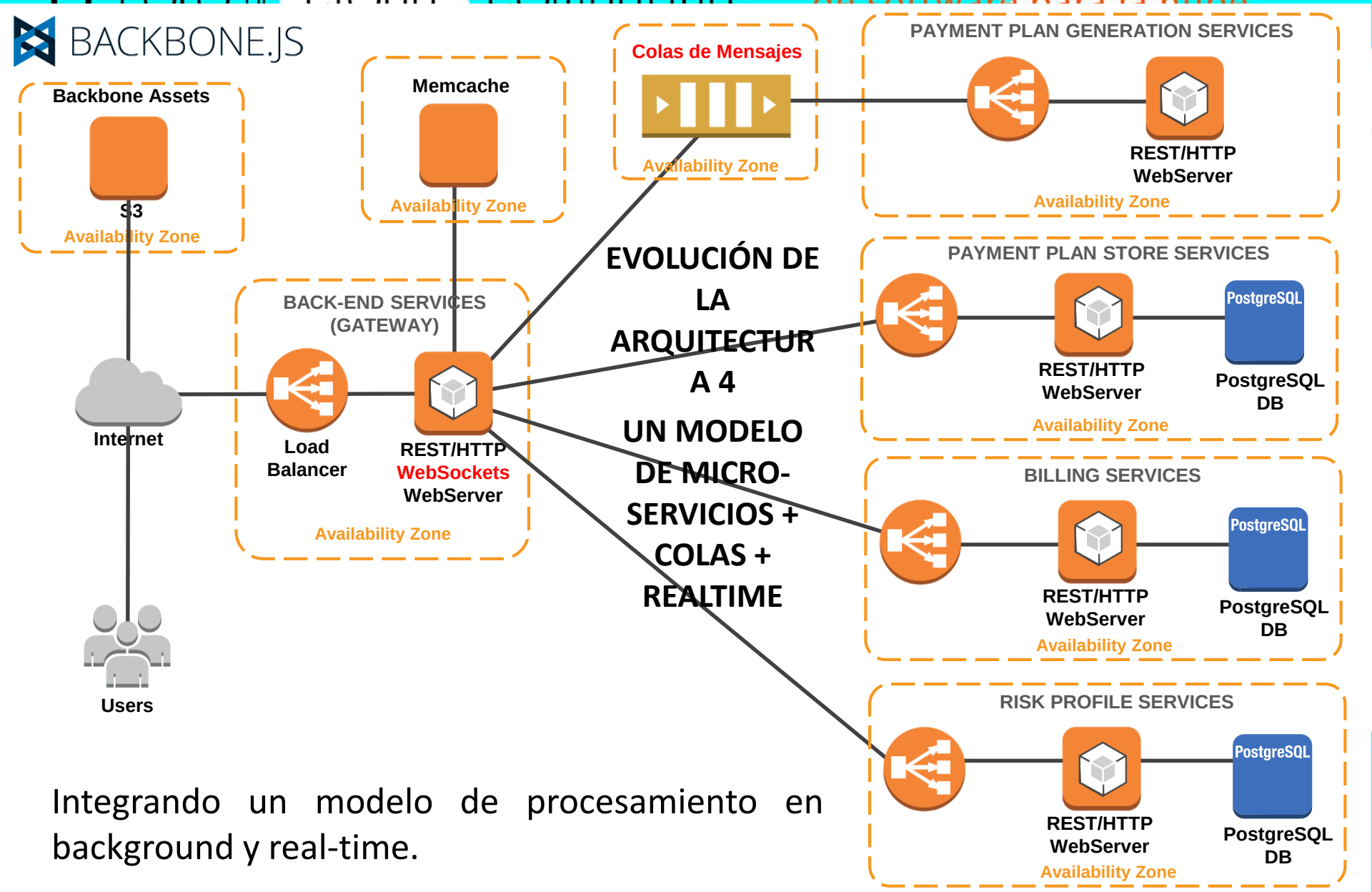
Migrando hacia un modelo de micro-servicios con Grails.



Migrando hacia un modelo de micro-servicios con Play/Scala.



Migrando hacia un modelo de micro-servicios con Play/Java.



Integrando un modelo de procesamiento en background y real-time.

```
name := "simbancariopaymentplansdb"

version := "1.0-SNAPSHOT"

libraryDependencies ++= Seq(
  jdbc,
  anorm,
  cache,
  "com.newrelic.agent.java" % "newrelic-agent" % "3.6.0",
  "com.typesafe.slick" %% "slick" % "2.0.2",
  "postgresql" % "postgresql" % "9.1-901.jdbc4",
  "com.h2database" % "h2" % "1.3.170",
  "com.github.tototoshi" %% "slick-joda-mapper" % "1.1.0"
)

play.Project.playScalaSettings

```

```
# Logger provided to your application:
logger.application=DEBUG

play {
  akka {
    akka.loggers = ["akka.event.slf4j.Slf4jLogger"]
    loglevel = WARNING
    actor {
      default-dispatcher = {
        fork-join-executor {
          parallelism-min = 1
          parallelism-max = 24
          parallelism-factor = 1
        }
      }
    }
  }
}
```

LECCIONES APRENDIDAS

- Servidor de aplicaciones
  - Manejo de threads
  - Gestión de respuestas HTTP
  - Gestión de conexiones a DB
  - Procesamiento de tareas intensivas en cómputo
- Framework de desarrollo
  - Configuración por defecto
  - Gestión de threads y paralelismo
  - Ejecución asíncrona de llamados REST
  - Manejador de persistencia (EclipseLink, Hibernate, Slick, ebeans, etc.)
  - Gestión de estados
  - Librerías utilizadas

Las labores de tunning de cada stack tecnológico y las pruebas de rendimiento consumen bastante tiempo.

LECCIONES APRENDIDAS

```
#Services for algorithms
GET     /algorithms                 controllers.Algorithms.index()
GET     /algorithms/:id             controllers.Algorithms.show(id: String)
GET     /algorithms/:id/detail      controllers.Algorithms.detail(id: String)
POST    /algorithms                 controllers.Algorithms.save()
DELETE  /algorithms/:id             controllers.Algorithms.delete(id: String)

#Services for credit lines
GET     /credit_lines               controllers.CreditLines.index()
GET     /credit_lines_memory        controllers.CreditLines.memory
GET     /credit_lines/:id           controllers.CreditLines.show(id: String)
GET     /credit_lines/:id/detail    controllers.CreditLines.detail(id: String)
POST    /credit_lines               controllers.CreditLines.save()
DELETE  /credit_lines/:id           controllers.CreditLines.delete(id: String)

#Services for payment plans
GET     /payment_plans                  controllers.PaymentPlans.index
GET     /payment_plans/delete_all       controllers.PaymentPlans.deleteAll
GET     /payment_plans/:id              controllers.PaymentPlans.show(id: String)
POST    /payment_plans                  controllers.PaymentPlans.save
GET     /payment_plans/users/:cedula    controllers.PaymentPlans.showOfUser(cedula: String)

#Services for tenants
GET     /tenants                controllers.Tenants.index()
GET     /tenants/:id            controllers.Tenants.show(id: String)
POST    /tenants                controllers.Tenants.save()
DELETE  /tenants/:id            controllers.Tenants.delete(id: String)
PUT     /tenants/:id/set_admin_user controllers.Tenants.setAdminUser(id: String)
```

- Servicios de DB
- Servicios para tareas intensivas en cómputo
- API aplicación web
- API aplicaciones móviles

Hacer un buen diseño de APIs REST para aplicaciones basadas en micro-servicios requiere un modelo de desarrollo y despliegue diferente.

```scala
object CreditLines2 extends Controller {

  def index = Action.async { implicit request =>
    val url_from_config = current.configuration.getString("url_simbancariopaymentplansdb")
    url_from_config match {
      case Some(url_host) =>
        for {
          response <- WS.url(url_host + "/credit_lines").
            withHeaders("Content-Type" -> "application/json").get()
        } yield {
          val responseWithFormat = "{\"credit_lines\":" + response.body + "}"
          val credit_lines = (Json.parse(responseWithFormat) \ "credit_lines").as[Seq[CreditLine]]
          Ok(Json.toJson(credit_lines))
        }
      case None =>
        scala.concurrent.Future(BadRequest("URL of the service NOT found"))
    }
  }
```


```java
public class CreditLines extends Controller {

    public static Promise<Result> index() {
        String url_from_config = Play.application().configuration()
                .getString("url_simbancariopaymentplansdb");
        if (url_from_config != null) {
            final Promise<Result> resultPromise = WS
                    .url(url_from_config + "/credit_lines").get()
                    .map(new Function<WS.Response, Result>() {
                        public Result apply(WS.Response response) {
                            JsonNode node = response.asJson();
                            return ok(node);
                        }
                    });
            return resultPromise;
        } else
            return F.Promise
                    .pure((Result) badRequest("URL of the service NOT found"));
    }
```

Nuevos frameworks como Play, lenguajes de back-end como Scala (por ser funcional) y modelos de actores basados en mensajes como Akka requieren una curva de aprendizaje considerables.

LECCIONES APRENDIDAS

BACKBONE.JS

ANGULARJS
by Google

Frameworks web MVC de lado del cliente (browser) como Backbone.js y Angular requieren de una curva de aprendizaje considerable y de nuevas habilidades por parte de los desarrolladores de Front-end (HTML, CSS y Javascript).

RETO 3

¿Cuál solución de IaaS/PaaS utilizar y cómo hacer uso eficiente de estás soluciones para desplegar las aplicaciones SaaS?

FRENTE DE TRABAJO 3

Evaluación y selección de proveedores IaaS/PaaS

USO DE SOLUCIONES IaaS/PaaS

- Plataformas cloud (PaaS) como Heroku proveen un rendimiento muy variable por lo cual medir el rendimiento en un elemento de la aplicación es complejo.
- Plataforma cloud (PaaS) como Heroku exponen todas las aplicaciones a Internet lo cual hace más complejo el despliegue de servicios privados (que no deben estar expuestos a Internet).
- El despliegue de aplicaciones sobre soluciones como Amazon Web Services requiere de tareas complejas de automatización para el despliegue de apps basadas en micro-servicios..

LAS PROBLEMÁTICAS ABORDADAS

RETO 4

¿Cómo aprovisionar las aplicaciones bajo el modelo de auto-servicio a través de un marketplace?

FRENTE DE TRABAJO 4

Desarrollo de un marketplace para el aprovisionamiento de aplicaciones

LAS PROBLEMÁTICAS ABORDADAS

RETO 5

¿Cuál debe ser el modelo de negocio y la estrategia de marketing para comercializar las aplicaciones?

FRENTE DE TRABAJO 5

Definición de un modelo de negocio y una estrategia de marketing

Harold Castro, Ph.D.
hcastro@uniandes.edu.co
Director del Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
Bogotá, Colombia

Universidad de los Andes

HEINSOHN MAPEO
MANEJO DE PROCESOS ORGANIZACIONALES

COLCIENCIAS
Departamento Administrativo de Ciencia, Tecnología e Innovación

RETOS	FRENTES DE TRABAJO
¿Cómo reducir los costos de desarrollo de aplicaciones individuales?	Entendimiento del dominio y desarrollo de una línea de productos bajo el modelo SaaS
¿Cómo desarrollar aplicaciones escalables bajo el modelo SaaS (arquitecturas, tecnologías, etc.) ?	Selección y evaluación de arquitecturas y tecnologías para construir aplicaciones escalables
¿Cuál solución de IaaS/PaaS utilizar y cómo hacer uso eficiente de estás soluciones para desplegar las aplicaciones SaaS?	Evaluación y selección de proveedores IaaS/PaaS
¿Cómo aprovisionar las aplicaciones bajo el modelo de auto-servicio a través de un marketplace?	Desarrollo de un marketplace para el aprovisionamiento de aplicaciones
¿Cuál debe ser el modelo de negocio y la estrategia de marketing para comercializar las aplicaciones?	Definición de un modelo de negocio y una estrategia de marketing