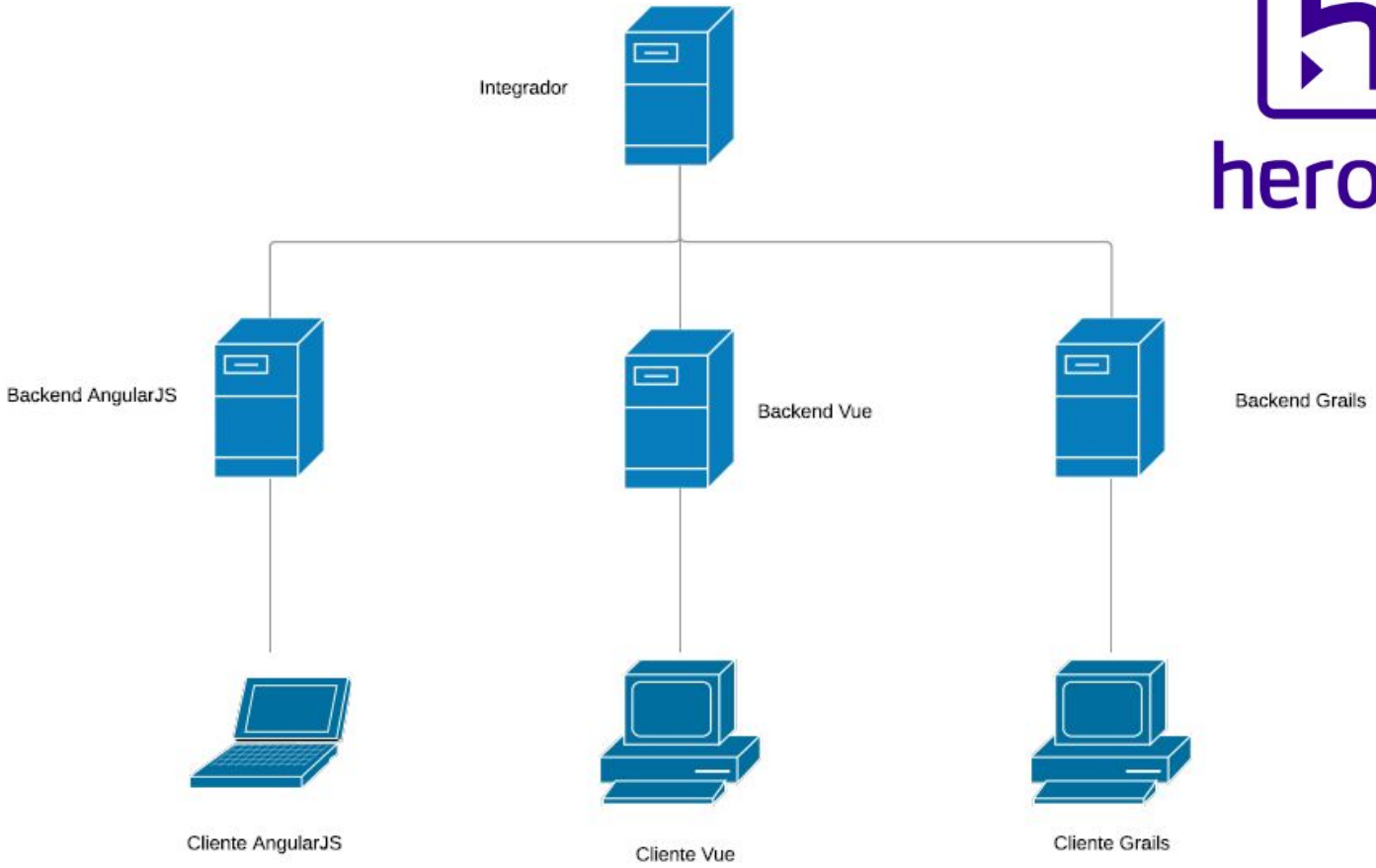
Arquitectura Web

Integración

Diagrama de componentes



heroku

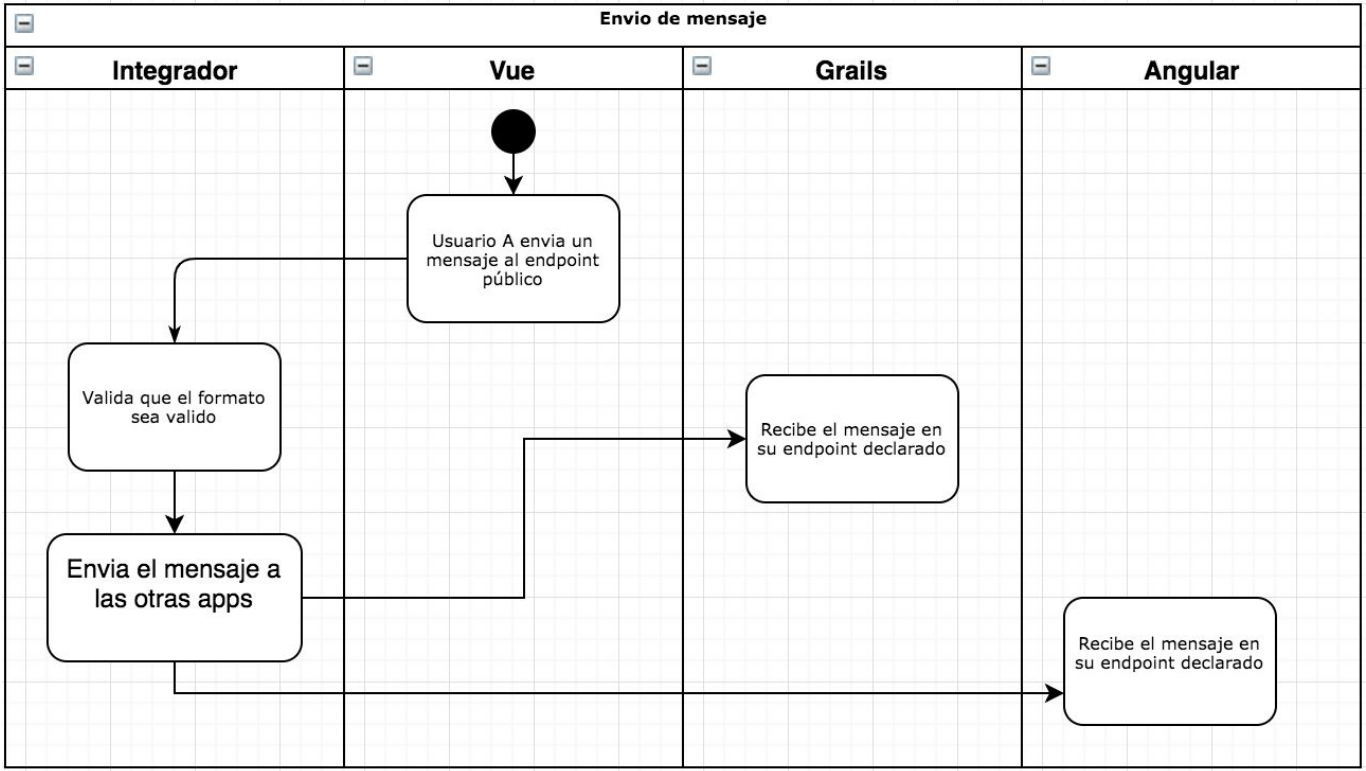
Definición del integrador

- Para la definición de la solución integral de manejaron varias primeras ideas de las cuales se seleccionó la de generar un integrador por fuera de los chats a integrar para dividir la responsabilidad de una manera más homogénea favoreciendo de esta forma también la integración de otra aplicación sin necesidad de adaptarse a una tecnología en particular.

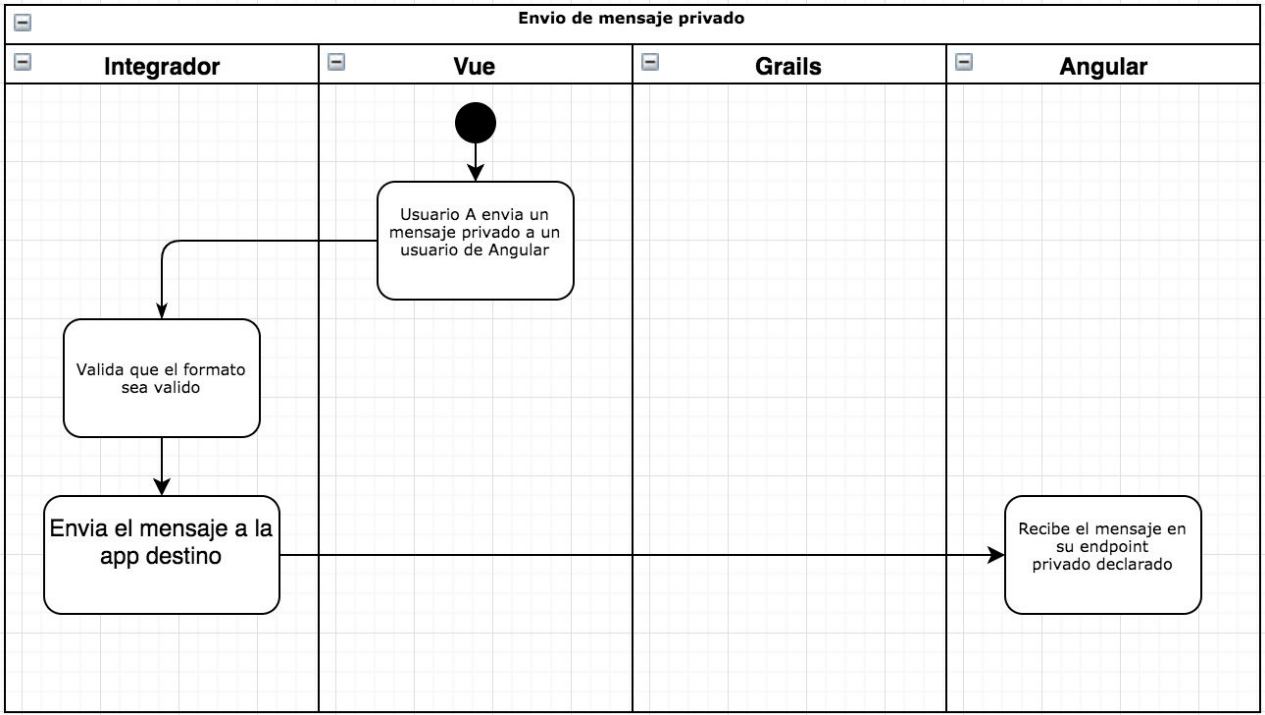
- Se definieron un conjunto de mensajes básicos que todas las aplicaciones registradas debían respetar:
  - Notificación de contactos en el chat
  - Mensaje a través del canal público
  - Mensaje a través del canal privado (las aplicaciones de chat que no lo soportan, los desestiman)
  - Envío de archivos de Imágenes y sonido (las aplicaciones de chat que no lo soportan, los desestiman)

- También se definió la mecánica de trabajo con el integrador y el formato de cada uno de los mensajes

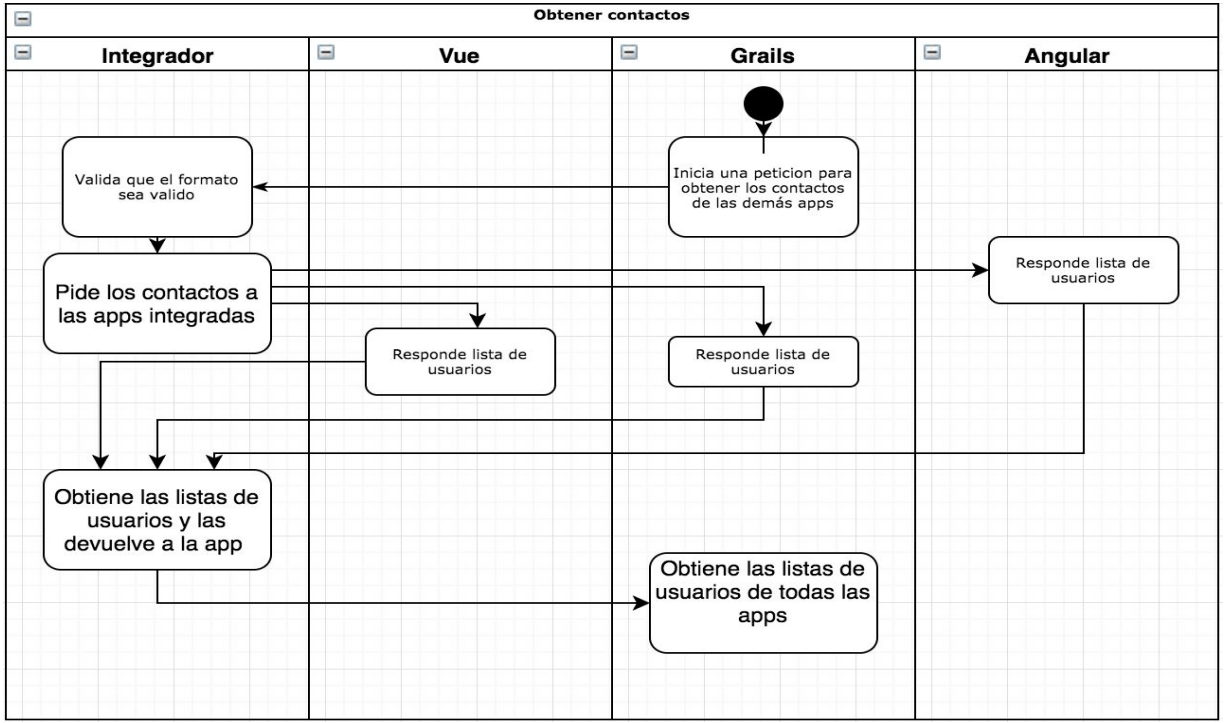
Envío de mensaje en canal público



Envío de mensaje privado



Obtener contactos



Integrador: Mecánica

1. Registración de una nueva app al integrador
   1.1. La nueva aplicación se registrará en el integrador enviando sus datos al endpoint '**/integrate**'.
   1.2. El integrador registra la información de la aplicación.
2. Propagación de contactos entre las aplicaciones
   2.1. La nueva aplicación pide los contactos al integrador.
   2.2. El integrador solicita los contactos a todas las aplicaciones registradas armando un arreglo.
   2.3. El integrador envía el arreglo al solicitante.
3. Envío de mensajes entre aplicaciones
   3.1. El usuario ‘Pablo’ (aplicación A), envía un mensaje ‘hola’ al usuario ‘Hernán’ que es usuario de otra aplicación (aplicación B).
   3.2. La aplicación A adapta el mensaje de un formato interno al formato genérico especificado.
   3.3. La aplicación A envía el mensaje al endpoint '**/private/send**' del integrador.
   3.4. El integrador usa la información en el mensaje para reenviarlo a la aplicación correspondiente.
   3.5. La aplicación B recibe el mensaje ‘hola’ en formato genérico y lo adapta a sus necesidades para direccionar a su usuario local ‘Hernán’.
   3.6. El usuario ‘Hernán’ recibe el mensaje de ‘Pablo’ como mensaje privado.

Integrador: Mensajes

Ejemplo de mensaje genérico esperado por las aplicaciones

```
{
"to": {"id": "kp9gu61Zv_dCYV2HAAAB","name": "alvaro"},
"from":{"id": 1, "name":"fulanito"},
"msg":"hola",
"attachment":null,
"sourceApp":"grails",
"targetApp": "AngularJSChatApp"
}
```

Ejemplo de mensaje de registración de una nueva aplicación hacia el integrador

```
{
    "id": "grails",
    "endpointPublic":"https://arq-chat.herokuapp.com/integrate/public",
    "endpointPrivate":"https://arq-chat.herokuapp.com/integrate/private",
    "endpointContacts":"https://arq-chat.herokuapp.com/show/users"
}
```

Listado de contactos provisto por el integrador en el endpoint '**/contacts**'

AngularJs: Cambios para adaptarse al integrador

Accion Server	TP1	TP2
Send Msg. Publico	Cli1 envía mensaje público el cual llega al back y utilizando los metadatos se envía al socketId del chat público el cual muestra el mensaje.	Cli1 envía mensaje público el cual llega al back, se envia el msg al socket del chat público, el cual lo adapta y envía al integrador como mensaje público, el cual se encarga de enviarlo a las otras aplicaciones.
Send Msg. Privado	Cli1 envia msg a Cli2, el msg llega al back y es enviado al socketid del Cli2 el cual recibe el mensaje.	Cli1 envia msg a Cli2, el msg llega al back y dependiendo de los metadatos appId + socketId determina si es un mensaje local con lo cual opera como en el caso anterior o si es un msg remoto en cuyo caso el mensaje es adaptado y enviado al integrador, el que se encarga de rutear el mensaje hacia la aplicación correspondiente.
Receive Msg. Publico	-	El server recibe un post http con el mensaje público. El server, busca el socketId del chat público y lo envía.
Receive Msg. Privado	-	El server recibe un post http con un mensaje privado. Toma los datos del mensaje, y lo asume como propio. Se busca el socketId del usuario para enrutar el mensaje hacia el.

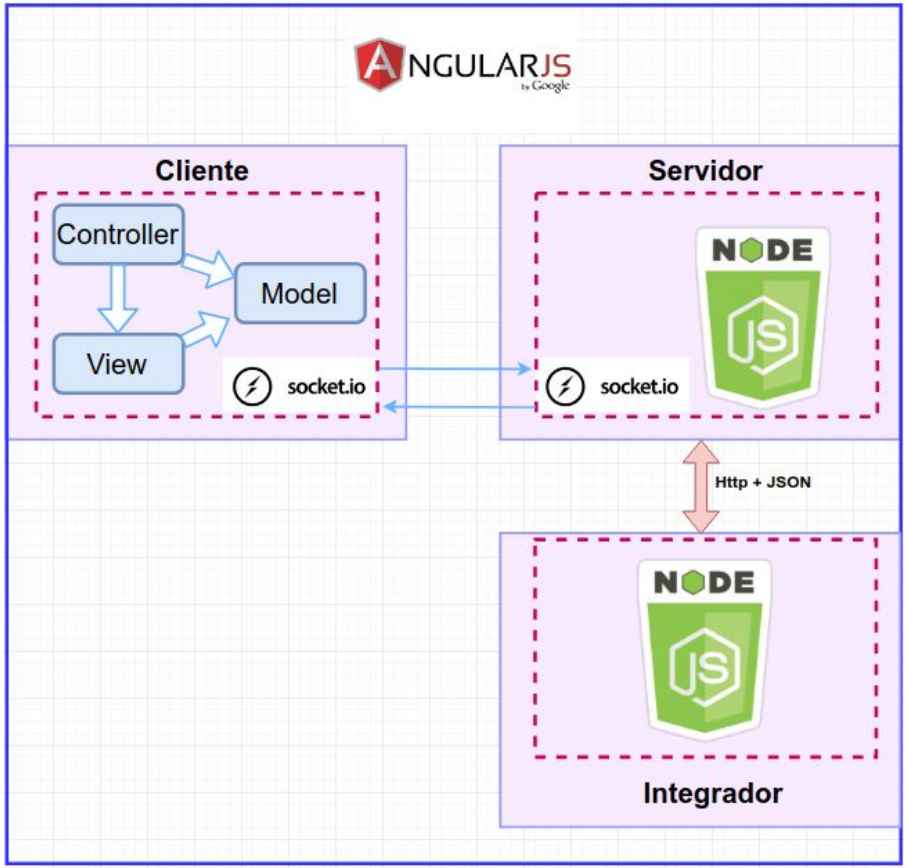
AngularJs: Cambios para adaptarse al integrador

- Se trabajó exclusivamente en el backend
  - Se agregaron los endpoints necesarios por el protocolo implementado en el integrador
    - **GET /contacts**: Para notificar el listado de usuarios de la aplicación
    - **POST /publicChat**: Para la recepción de mensajes públicos
    - **POST /privateChat** Para la recepción de mensajes privados
  - Se hicieron modificaciones sobre la mecánica de envío y recepción de mensajes y sonido
  - Se hicieron modificaciones para incluir usuarios de aplicaciones externas
  - Se incluyó un adaptador para la traducción de mensajes recibidos de otras apps a través del integrador y para los mensajes enviados al integrador.

```
var adapter = {
  adaptToExternalMessage: function(msg){
  adaptToExternalFile: function(msg){
  adaptToInternalFile: function(msg){
  adaptToInternalMessage: function(msg){
  adaptToExternalContact: function(contact){
  adaptToInternalContact: function(externalContact, externalAppName){
  adaptToInternalContacts: function(externalContacts){
};
```

AngularJs: Nuevo feature

AngularJs: Diagrama de despliegue



Cambios para adaptarse al integrador: Vue

Para la adaptación del chat a la integración con el traductor fue necesario el desarrollo de una serie de endpoints para la correcta comunicación con el mismo.

- **GET /api/contacts**: Para notificar los usuarios nativos de la aplicación.
- **POST /api/public/send**: Para la recepción de mensajes del channel público
- **POST /api/private/send** Para la recepción de mensajes privados de usuarios

Mensajes creados en Vue

La otra parte importante que se modificó fue en la lógica de mensajes del chat.

- Si se escribe un mensaje en el chat público:
  - Se emite el mensaje por el socket a todos los usuarios para el canal global.
  - Se envía mensaje al integrador

- Si se recibe un mensaje privado para un destino Vue:
  - Se emite el mensaje por el socket para el usuario destino

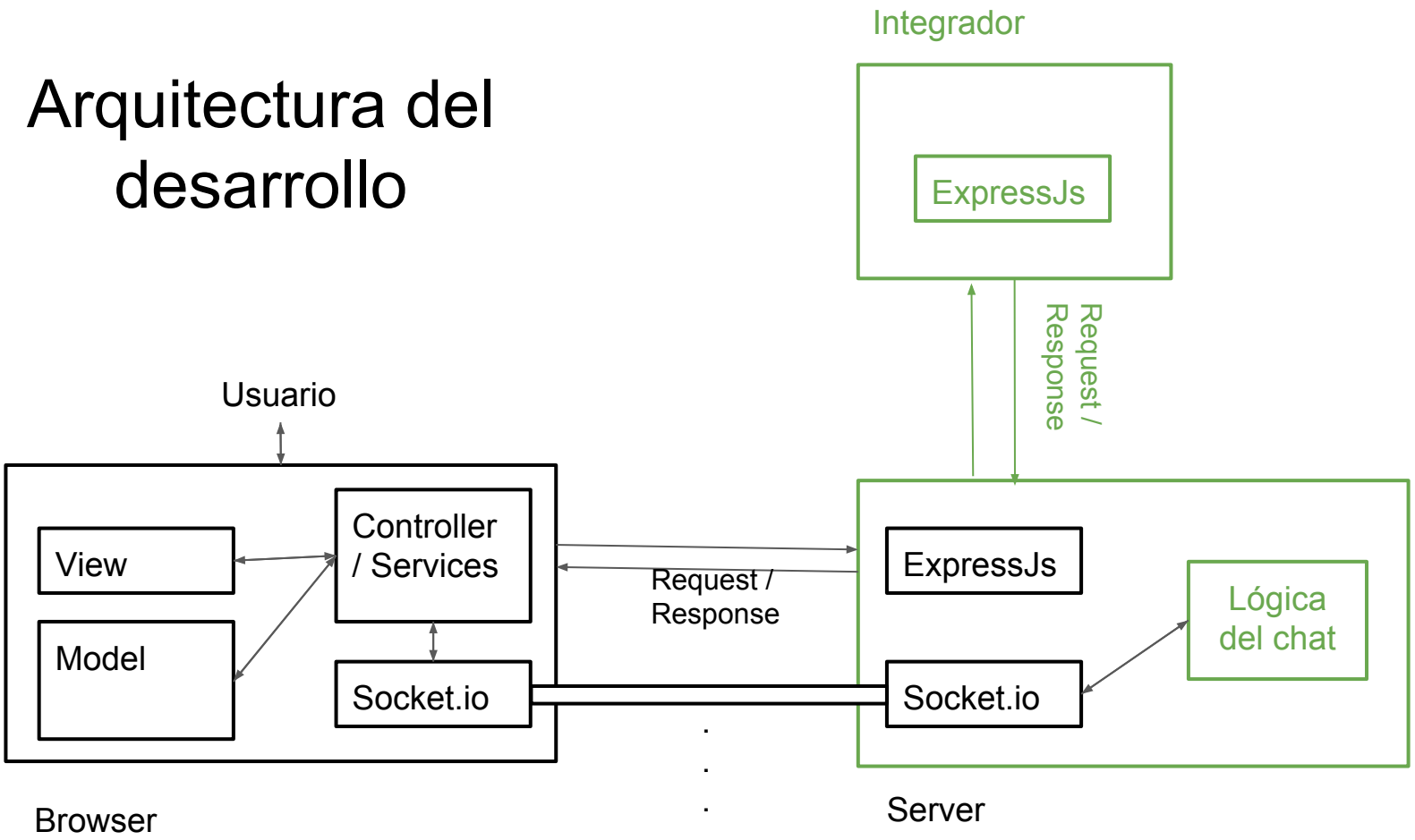
- Si se recibe un mensaje privado para un destino distinto de Vue:
  - Se emite el mensaje para el usuario destino
  - Se envía mensaje al integrador.

Mensajes creados en Vue

- Si se recibe un mensaje en **/api/public/send**:
  - Se registra el usuario externo y se le asocia la app origen
  - Se emite el mensaje por el socket a todos los usuarios para el canal global.

- Si se recibe un mensaje **/api/private/send**:
  - Se registra el usuario externo y se le asocia la app origen
  - Se emite el mensaje por el socket para el usuario destino

Arquitectura del desarrollo



Cambios para adaptarse al integrador: Grails

Se agregaron 3 endpoints

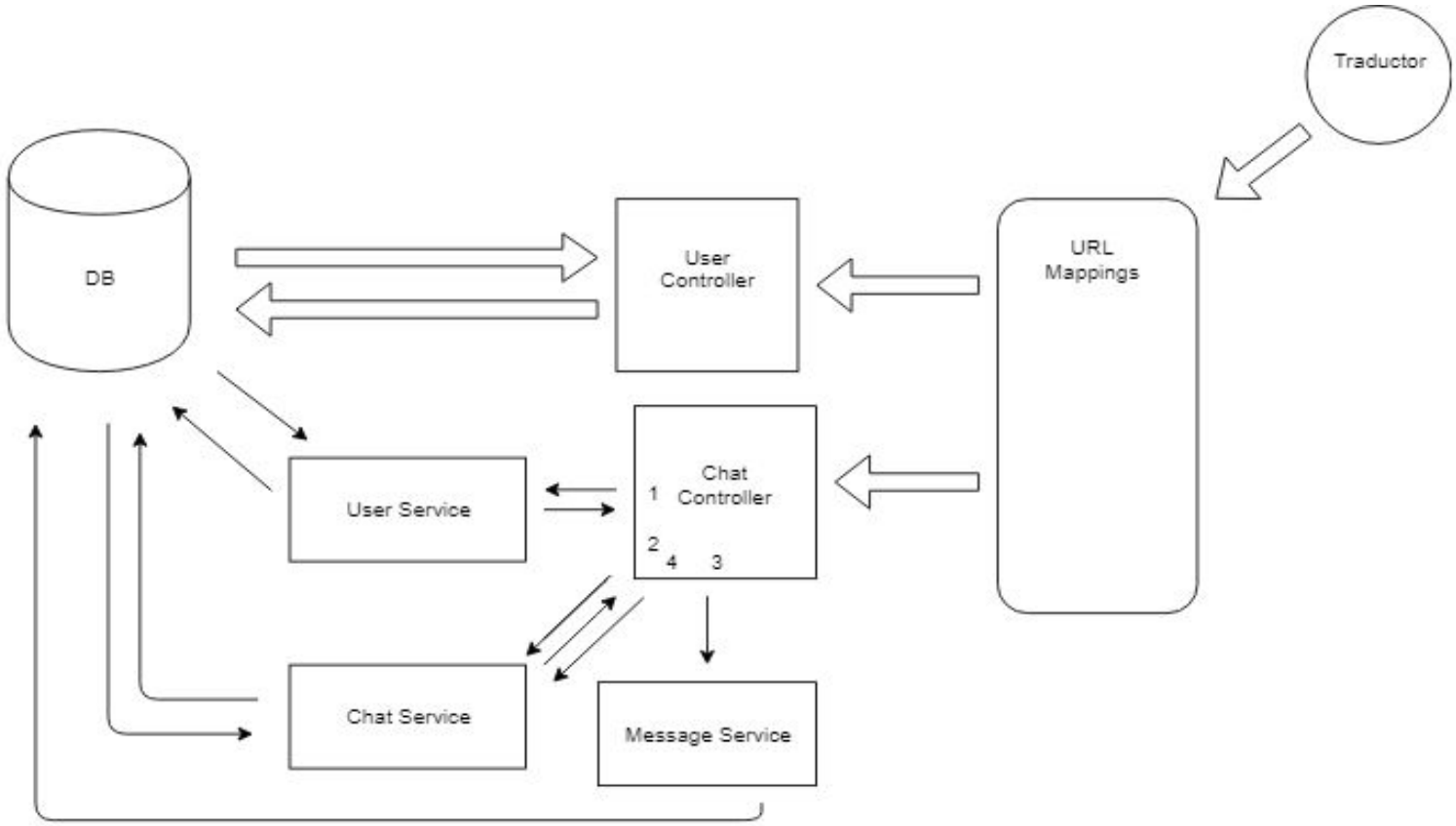
- /show/user
  - Recibe gets
  - Respondido por userController -> userService ->db -> vista
  - Retorna una lista de usuarios
- /integrate/public e /integrate/private
  - Reciben posts
  - Son atendidos por chatController -> userService (buscar / crear usuario externo) -> chatService (buscar / crear chat) -> mesaggeService (crear y agregar msg al chat) -> chatService (avisar sobre el msg)

Se agregaron 3 parámetros más a la clase usuario (se ven reflejados como columnas en la tabla users):

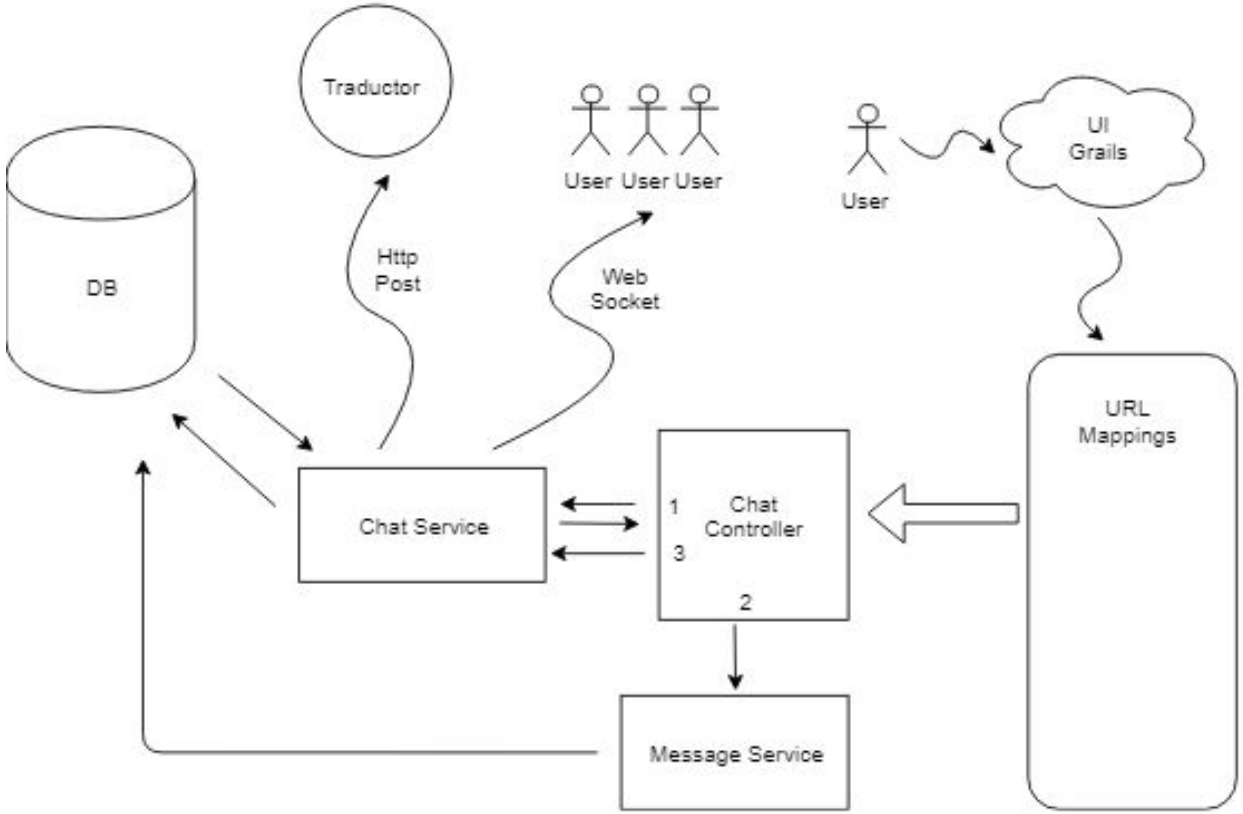
- String IntegrationApp
- String IntegrationID
- Enum Type (LOCAL / INTEGRATION)

Se agregó una pestaña en la que se muestra todos los usuarios externos, consume el endpoint del traductor que muestra los usuarios de los otros chats. Filtramos los nuestros

Ingreso de mensajes vía traductor



Envío de mensajes al traductor



Demo

Requerimientos para una cuarta aplicación a integrarse

Cualquier chat que se quisiera integrar a la plataforma debería cumplir los siguientes requisitos:

- registrarse en el endpoint de integrate notificando los 3 endpoint a través de los cuales se le realizarán las notificaciones de los mensajes (mensajes públicos, privados y sus contactos).
- notificar de los mensajes realizados en el chat público y en los privados con contactos de otras aplicaciones al integrador a través de los endpoints correspondientes en el formato json.

Comentarios

Ventajas:

- Fácil de integrar a una nueva aplicación
- Solo se carga el traductor cuando son mensajes cross aplicación (los mensajes privados dentro de las aplicaciones no usan el integrador)

Desventajas:

- No tiene seguridad
- No persiste información de las aplicaciones que la componen
-

Referencias

- Repo AngularJS: https://git.exactas.uba.ar/ichiapella/arqWeb18
- Repo Grails: https://github.com/jnvillar/arqWeb-translator
- Repo Vue: https://github.com/mferranti/arquitecturaweb
- Repo Integrator: https://github.com/mferranti/arquitecturaweb-integration

- Documentación de los endpoints del integrador: https://awebchat-integration.herokuapp.com/

- Chat AngularJS: https://quiet-citadel-52217.herokuapp.com/
- Chat vue: https://awebchat.herokuapp.com/chat
- Chat Grails:https://arq-chat.herokuapp.com/login

¿Dudas, consultas?

¡Muchas Gracias!