

Micro-services  
Here we go!

# Assumptions

We are looking to migrate BEEP to a microservices architecture.

The specifications are the same as described during all previous iterations of beep.

You will need to answer 10 questions:

- Via the redaction of the TAD (components architecture, sequences diagram, deployment architecture, etc.)
- Via the production of POCs (when mentioned)
- For your schema, use draw.io
- Redact your report using asciidoc

You are forbidden to propose or consider the following technologies:

- Messages queue
- CQRS
- Event Sourcing

# Q1: Starting with the prerequisites, what do you propose to separate your application into functional neighborhoods and microservices?

Redefine all functionalities in the form:

- “As [guest user / server admin / ...], I want to [create a channel / ...] so that [the user can self-assess his channel / ...]”
- Organize these proposals into coherent functional neighborhoods
- Propose an architecture diagram for the breakdown of your application into (functional) microservices.

Help resources:

- <https://martinfowler.com/bliki/BoundedContext.html>
- <https://martinfowler.com/articles/break-monolith-into-microservices.html>
- <https://leofvo.me/articles/microservices-for-the-win>

# Q2: How do I manage the authentication system with an OIDC?

Draw an architecture diagram of your application

Present the deployment diagram for your Beep application and the OIDC

Present sequence diagrams of the following actions:

- User creates Beep account (vanilla)
- User creates Beep account (via Polytech account)
- User creates Beep account (via Google account)
- User logs in (vanilla)
- User logs in (Polytech)
- User associates their Google account with their Beep account

Constraints:

- OIDC will be Keycloak
- A user can log in with his Polytech account (=> Polytech LDAP access via OIDC - to be taken into account in your deployment scheme)
- A user can associate his user account with a Google account
- Make a POC by modifying the current Beep, and implementing the following cases: vanilla account creation, Google account creation, vanilla connection, Google connection.
- POC delivery: deployment guide, source code + functional deployment on your server (give login url)

## Q3. Inter microservices communication

How will you orchestrate inter-microservice communication? API systems?  
Protocols used, etc.

- Explain your approach
- Make a sequence diagram of the communication between some of your services to present your approach.
- Make a POC of communication between two microservices (helloworld) using the system you intend to implement.

# Q4: How would you implement an authorization system ?

Define what is a authorization service.

Define the technical and functional architecture and the technologies you recommend.

Present different sequence diagrams for the application's main actions.

**Note:** Your solution should enable you to manage permissions by server, category and channel.

You'll also need to set global permissions for platform administrators.

## Q5. How can I trace logs and queries?

We want to be able to observe the system's behavior in response to a user request.

- Define the system and the technical components to be implemented.
- Draw a deployment diagram of the various elements
- Draw a sequence diagram of a query (of your choice) to illustrate how your system works.
- Describe the needs of “security based logs”, to plug your system into a managed SOC

## Q6: Production ready system

Describe in a detailed matter how do you manage :

- Data security, data backup and restore
- Observability, and services supervision integrated to an existing enterprise system
- Infrastructure high disponibility, continuity plans

Draw up diagrams, in each of the cases

Draw up a target diagram



## Q7: Infrastructure security

> How to add a mobile application to the system

*Draw a sequence diagram of the mobile application's authentication process.*

> How do you implement the physical architecture and the network to meet state of the art separation between security zones (DMZ ...)

> How are managed cryptography, certificates, which protocols do you choose and why

## Q8: How do I add a search engine?

Make a functional and technical proposal for the search engine.

We want the user to have a full-text indexing engine.

For example, a user typing the keyword “rabbit” (but this could be a string of words) should have all messages, etc. containing this keyword brought up in a user interface.

Make a UI mockup (sketch) of the different phases of the use case.

Propose a technical stack and the indexing and search sequence diagram.

## Q9. How to manage platform security issues

Analyze how you secure the various components of your architecture (network, microservices, etc.).

How do you secure communications between microservices?

Present your security proposal, using sequence diagrams to show how your system will work.

Make a Poc of securing helloworld microservices with each other and with the outside world.

## Q10. How to integrate UI applications ?

How do you integrate your microservices together to present a good user experience?

Take the case of video streaming (conferences) and show how to integrate it correctly without the user thinking it's a separate application.

Make proposals with the necessary diagrams,

Your POC must be able to show that this has not changed the user experience.

# TimeLine

**3 february:** Goals announcement

**30 May 23h42:** send git repo containing

- The asciidoc file + pdf version
- Images (png + drawio source)
- The folder contains the production urls for your pocs and the related branches or repositories

Intermediate date:

**5 march:** presentation of a draft of your report

- Show asciidoc template in place,
- drafts of answers to questions (30-50%), images, etc.

**30 march:** complete V0 file

- answers to all questions (100%)
- Diagrams available (80%)
- POCs available (50-70%)

# Goals announcement

The aim of this iteration is to prepare the implementation of the microservice architecture for your application.

Now that the monolith is getting too big, it's time to slice up the application. However, the change of architecture requires the addition of technical layers to ensure maintenance and smooth operation.

- > 1 Technical Architecture Document per person. (English or French)

- > Use github for deliveries. (add leofvo & MonstyFred)

Of the 10 questions, you'll have to choose 8. (*bonus if all covered*)

Questions 1,2,3,4,5,6 are mandatory.

# Resources

# Exemple de DAT niveau “ light” (premier jet)

1	<u>PRESENTATION DU SOCLE TECHNIQUE CIBLE</u> .....	2
2	<u>ARCHITECTURE TECHNIQUE</u> .....	2
2.1	<u>Principes généraux</u> .....	2
2.2	<u>Cartographie et documentation</u> .....	3
2.3	<u>Spécificités OSHIMAE</u> .....	4
2.4	<u>Zones logiques des fonctions du paysage système</u> .....	4
2.5	<u>Paysages systèmes par projet</u> .....	4
2.6	<u>Cloisonnement et segmentation des réseaux</u> .....	5
2.7	<u>Stockage</u> .....	7
2.8	<u>Supervision</u> .....	7
2.9	<u>Sauvegarde</u> .....	7
2.10	<u>Plan de reprise informatique</u> .....	8
2.11	<u>Evolutivité</u> .....	8
2.12	<u>Élasticité</u> .....	8
2.13	<u>Volumétrie initiale</u> .....	8
3	<u>INFRASTRUCTURE</u> .....	8
3.1	<u>Services</u> .....	8
3.2	<u>Réseau WAN</u> .....	9
3.3	<u>Système d'exploitation et centriciels (Middleware)</u> .....	9
3.4	<u>Principes</u> .....	9
3.5	<u>Passerelles</u> .....	10

Grosso modo ca fait 15  
Pages grand max



# Exemple de DAT niveau “ avancé” (premier jet)

Objectif du document .....	1
Lexique .....	2
Contexte .....	5
Rappel du contexte .....	5
Rappel du périmètre .....	5
Historique des versions .....	5
Documents de références .....	5
Architecture Technique Générale .....	6
Présentation .....	6
Schéma global de l'architecture .....	7
Mise en haute disponibilité de la passerelle <u>RadosGW</u> .....	8
Plate-forme matérielle .....	9
Plate-forme technique .....	10
Flux réseau .....	11
Tableau des flux .....	12
Cartographie des flux .....	12
Authentification et autorisation .....	13
Implémentation de la solution .....	14
Implantation physique .....	14
Implémentation du réseau .....	16
Configuration des VLANs .....	16
Implémentation des services .....	18
Intégration avec « <u>Kubernetex</u> » .....	23
Composant 1 .....	
Configuration d'apache2 / <u>FastCGI</u> avec <u>Openssl</u> .....	23
Configuration de <u>Ceph</u> objet Gateway .....	24
Mise en haute disponibilité des 3 nœuds <u>RadosGW</u> .....	25
Configuration de <u>RadosGW</u> avec <u>Openstack</u> « <u>Keystone</u> » .....	26
Composant 2 .....	27
Configuration des pools pour les services « <u>Openstack</u> » .....	27
Configuration de l'authentification avec « <u>Cephx</u> » .....	27
Configuration des services <u>Openstack</u> pour « <u>Ceph</u> » .....	27
Sauvegarde de la solution .....	28
Copie de récupération rapide .....	28
Copie d'archive .....	29
journalisation de la solution .....	30
Monitoring de la solution .....	32

Grosso modo ca fait 30 pages  
et on est sur des règles  
complexes de mise en oeuvre

# Exemple de DAT niveau “ très (trop complet) ”

1	Contexte.....	5
1.1	Objet du document.....	5
1.2	Périmètre.....	7
1.3	Documents applicables et de références.....	8
1.3.1	Documents applicables.....	8
1.3.2	Documents de références.....	9
2	Exigences et contraintes.....	10
2.1	Performances et ergonomie.....	10
2.1.1	Connecteurs.....	10
2.1.2	Intranet Agricol.....	10
2.1.3	Annuaire.....	11
2.2	Haute disponibilité.....	11
2.3	Mutualisation et banalisation des plates-formes.....	11
2.4	Sécurité.....	12
2.4.1	Sécurité du système d'exploitation.....	12
2.4.2	Chiffrement des flux.....	12
2.4.3	Certificats serveurs.....	12
3	Les principes d'architecture.....	15
3.1	Fonctionnement de la réplication LDAP.....	15
3.2	Stratégie de haute-disponibilité.....	15
3.2.1	Rappels sur le protocole LDAP.....	15
3.2.2	Utilisation d'annuaires LDAP en mode multi-maîtres.....	16
3.2.3	Tolérance de panne.....	16
3.2.4	Équilibrage de charge.....	16
3.2.5	Fonctionnement de HeartBeat et HAProxy.....	17
3.3	Synthèse des principes retenus.....	21
3.3.1	Haute-disponibilité des machines.....	21
3.3.2	Haute-disponibilité des services.....	21
3.3.3	Périmètre d'utilisation de HAProxy et de Apache.....	21
3.3.4	Partage de la session sur le proxy d'authentification.....	22
3.3.5	En synthèse.....	22
3.4	Organisation réseau.....	22
3.5	Organisation des données.....	23
3.6	Organisation des utilisateurs systèmes.....	24
4	Architecture générale.....	26
4.1	Liste des composants.....	27
4.1.1	Nœud Annuaire Maître.....	27
4.1.2	Nœud Annuaire Esclave.....	27
4.1.3	Nœud Administration.....	27
4.2	Descriptif des composants.....	28
4.2.1	LinID Directory Manager (DM).....	28
4.2.2	OpenLDAP.....	29
4.2.3	LinID OpenLDAP Manager (OM).....	30
4.2.4	LDAP Synchronization Connector (LSC).....	30
4.2.5	Caractéristiques de LemonLDAP:NG.....	31
4.3	Dimensionnement physique des plates-formes.....	32
4.4	Dimensionnement des serveurs Tomcat.....	33

4.5	Les flux de la solution d'annuaire.....	34
4.5.1	Réseau d'accès au Datacenter.....	34
4.5.2	Au sein de la solution.....	34
4.5.3	En interface avec la solution.....	37
5	Sauvegarde / Restauration.....	39
5.1	Bonnes pratiques.....	39
5.2	Synthèse.....	39
6	Supervision et logs.....	40
6.1	Sondes à activer sur le serveur Centreon.....	40
6.2	Détails des sondes d'un serveur LDAP maître.....	40
6.2.1	Réseau.....	40
6.2.2	Système.....	41
6.2.3	SSH.....	41
6.2.4	Apache/httpd.....	42
6.2.5	Tomcat.....	43
6.2.6	OpenLDAP/slaped.....	43
6.3	Détails des sondes d'un serveur LDAP esclave.....	45
6.3.1	Réseau.....	45
6.3.2	Système.....	45
6.3.3	SSH.....	46
6.3.4	OpenLDAP/slaped.....	46
6.4	Détails des sondes d'un serveur LDAP maître.....	47
6.4.1	Réseau.....	47
6.4.2	Système.....	48
6.4.3	SSH.....	48
6.4.4	Apache/httpd.....	48
6.4.5	Tomcat.....	49
6.4.6	LSC.....	50
6.5	Gestion des logs.....	50

l'idée sera de  
complexifier  
l'environnem  
ent sur la  
3em itération