BlablaMove - Équipe C - Scope du projet

Week 39-40

Personas

- Mathieu, Admin infrastructure (Monitoring système)
- Hugo, Marketing (Monitoring vente/transaction)
- Jean, Support team (Monitoring système + ligne de vie des livraisons)
- Alex, Client final (disponibilité des services, ligne de vie des livraisons (vue plus limitée))

Use cases (triés par priorité)

Mathieu - Admin infrastructure

En tant que Admin Infrastructure**, je souhaite** visualiser les heures de pointe**, afin de** dimensionner mon système d'information.

En tant que Admin Infrastructure**, je souhaite** visualiser l'état de chaque service**, afin de** savoir si un problème se pose au niveau de l'infrastructure.

En tant que Admin Infrastructure**, je souhaite** recevoir un email lorsqu'un service n'est plus disponible**, afin de** prendre des actions de mitigation.

Hugo - Marketing

En tant que Marketing**, je souhaite** connaitre les métriques concernant le taux d'activité du site web**, afin de** connaître l'impact des campagnes publicitaires.

En tant que Marketing**, je souhaite** connaitre le nombre d'objets ayant transité sur le réseau**, afin de** faire une campagne sur la popularité grandissante de la plateforme.

En tant que Marketing**, je souhaite** connaitre les villes et routes les plus populaires**, afin de** cibler les zones sur lesquelles je dois accentuer la communication.

Alex - Client

En tant que Client final**, je souhaite** connaître l'état des services**, afin de** savoir si je peux utiliser la plateforme dans de bonnes conditions.

En tant que Client final**, je souhaite** avoir une vue sur le cycle de vie de la livraison**, afin de** savoir où sont mes objets.

En tant que Client final**, je souhaite** voir les objets en transit**, afin de** planifier la réception de mes objets pour mon déménagement.

Jean - Support team

En tant que Support Team**, je souhaite** avoir une vue sur le cycle de vie des livraisons plus détaillée que celle du client**, afin de** l'informer de potentiels problèmes sur l'acheminement d'un produit.

En tant que Support Team**, je souhaite** avoir accès à l'historique d'un utilisateur**, afin de** pouvoir statuer sur un litige.

En tant que Support Team**, je souhaite** connaître l'état des différents services**, afin de** pouvoir répondre le plus précisément aux clients qui m'appellent pour se plaindre d'un problème avec la plateforme.

Fonctionnalités du dashboard que l'on compte couvrir

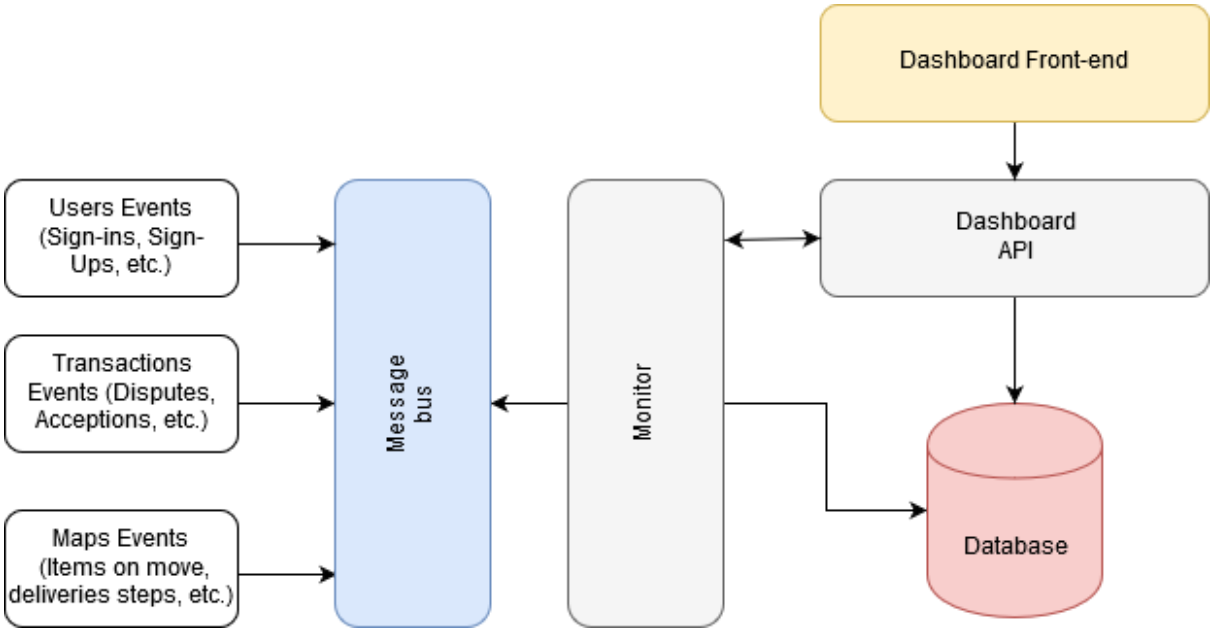
- **Orientées métier**
  - Transaction
    - Nombre d'acceptations de livraisons sur le temps d'une journée (i.e Les acceptations sont-elles plus nombreuses pendant l'heure du déjeuner)
    - Litiges (nombre, statut)
    - Avoir une ligne de vie avec l'état d'un objet à l'instant T, avec date de prise en charge et date estimée de livraison. S'inspirer de UPS ou ligne de vie Amazon.
  - Items
    - Prochains types d'items à être déplacés
    - Items déclarés perdus
    - Nombre d'items déplacés au total (et en moyenne sur un trajet)
  - Map
    - Items en transit
    - Routes (KPI distance parcourue en moyenne, routes les plus utilisées, villes les plus fréquentées)
  - Utilisateurs
    - Nombre d'utilisateurs en ligne sur une journée (graphe de fréquentation)
    - Taux d'inscription sur le temps
    - KPIs: Compte avec le plus grand nombre de points, moyenne de points transférés par utilisateur
- **Système**
  - Charge serveur
  - Erreurs de requêtes

- Taux de disponibilité
- Utilisation de la base de données
    - Écriture / Lecture
    - Capacité stockée

Fonctionnalités que l'on ne compte pas couvrir

- Temps moyen de chargement d'une page
- Temps moyen d'un utilisateur sur le site
- Nombre de serveurs en ligne

Architecture possible:



Week 41 (Rendu Mercredi 10 Octobre)

Scénarios choisis pour le POC

Cas d'utilisation: Connaître l'état du service
Acteur: Utilisateur du site Blablamove
Pré-conditions: L'utilisateur a l'impression que l'application mobile met beaucoup de temps à réagir.
Post-conditions: L'utilisateur est informé de la disponibilité des services
Scénario:

1. Il ouvre son navigateur et se rend sur la page du site: "Is Blablamove down ?".
3. Le site affiche la dernière mise à jour de la vérification d'activité, le taux de disponibilité et les derniers problèmes rapportés sous forme de graphique simple.
4 (optionnel). Il peut décider d'appuyer sur un bouton pour signaler des lenteurs sur le site.
Annexe:
Même style que ceci: https://downdetector.com/status/steam

Cas d'utilisation: Visualiser les heures de pointe
Acteur: Administrateur Infrastructure de Blablamove
Pré-conditions: Aucune
Post-conditions: L'administrateur dispose des taux d'utilisation du site en fonction du temps
Scénario:

1. L'administrateur veut vérifier la fréquentation du site sur les 24h dernières heures
2. Il ouvre son navigateur et se rend sur la page du dashboard interne à Blablamove
3. Le site affiche un graphique rapportant la fréquentation du site heure par heure avec le nombre d'utilisateurs sur le site

Cas d'utilisation: Visualiser les villes plus actives
Acteur: Hugo responsable marketing de Blablamove
Pré-conditions: Hugo souhaite connaitre les villes les plus actives afin de mieux cibler sa prochaine campagne de pubs
Post-conditions: Hugo a les statistiques d'activités par ville et peut détecter les lieux nécessitant un appui publicitaire
Scénario:

1. Hugo consulte la carte de chaleurs (gradient vert-rouge) afin de détecter les zones géographiques en perte d'activité
2. Hugo sélectionne un niveau d'activité
3. Hugo récupère les statistiques par lieu.

Roadmap jusqu'au POC

Week 41 - Semaine 8 Octobre

- Implémentation d'un walking skeleton (Du front jusqu'aux services générant des données à monitorer en passant par la persistance)

Week 42 - Semaine 15 Octobre

- Implémentation du premier scénario
- Mise en place d'un pipeline d'intégration

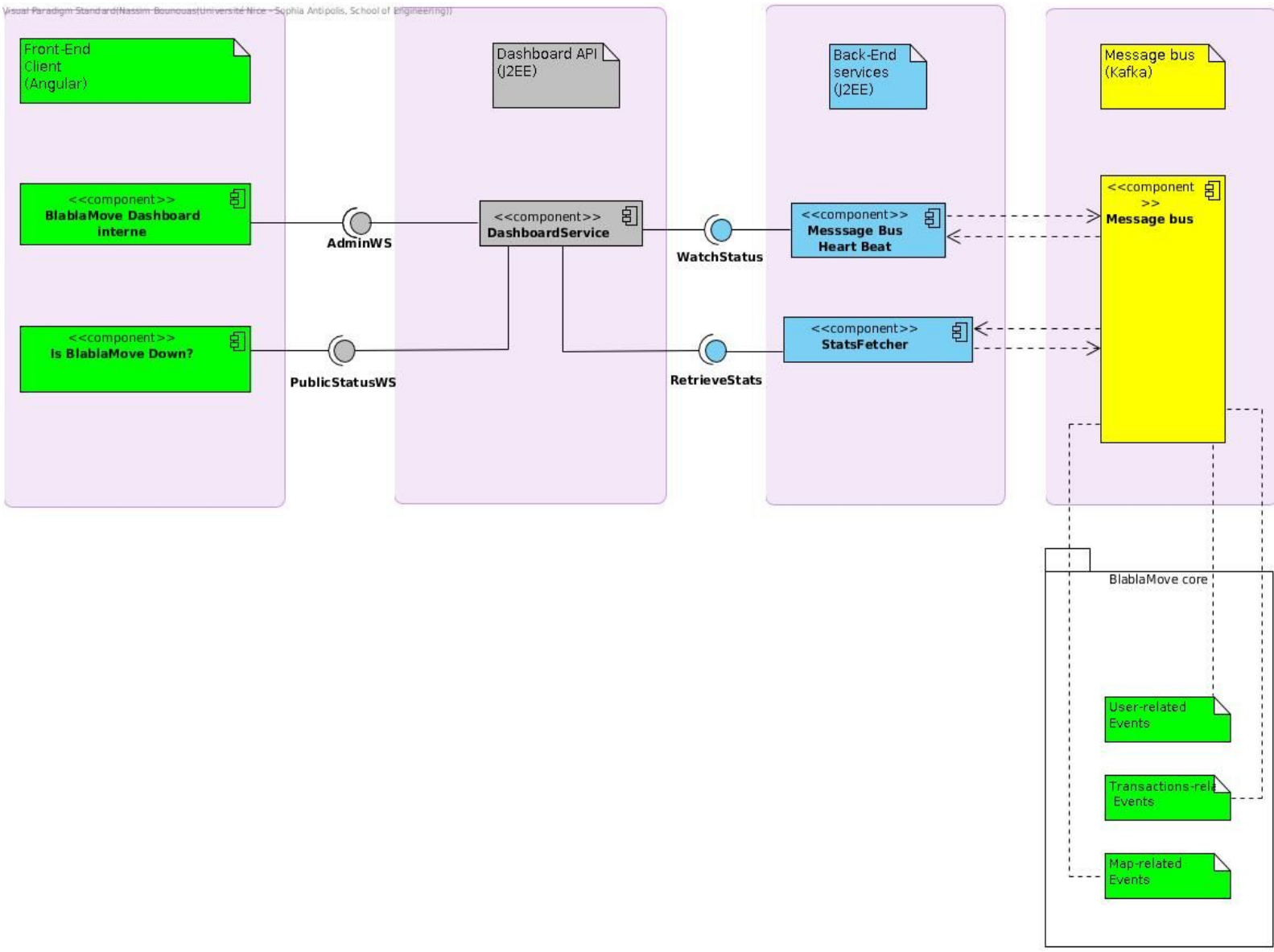
Week 43 - Semaine 22 Octobre

- Tests d'intégrations
- Implémentation du second scénario

Week 44 - Semaine 29 Octobre

- Verification, tests de charge et ajustements des interfaces

Spécification de l'architecture:



Interfaces

- **AdminWS**
  - int[ ] getVisitorNumbersPerHour()
  - LinkedList<Tuple<String, Int>> getMostActiveCities(); //(String=city, Int=transactions made in this city)
- **PublicStatusWS**
  - Date getLastCheckUpdate()
  - float getAvailabilityRate()
  - int[] getIssueNumbersPerHour()
- **RetrieveStats**
  - int NumberOfConnections(Date from, Date to);
  - int NumberOfTransactions(Date from, Date to);
  - List<Tuple<Date, Dispute>> RetrieveDisputes(Date from, Date to);

    - List<Transaction> RetrieveLastTransactions(String city, Date from);
    - int NumberOfConnectedUsers();
- **WatchStatus**
    - int[] TriggerHeartBeat(); //Temps de réponse en ms

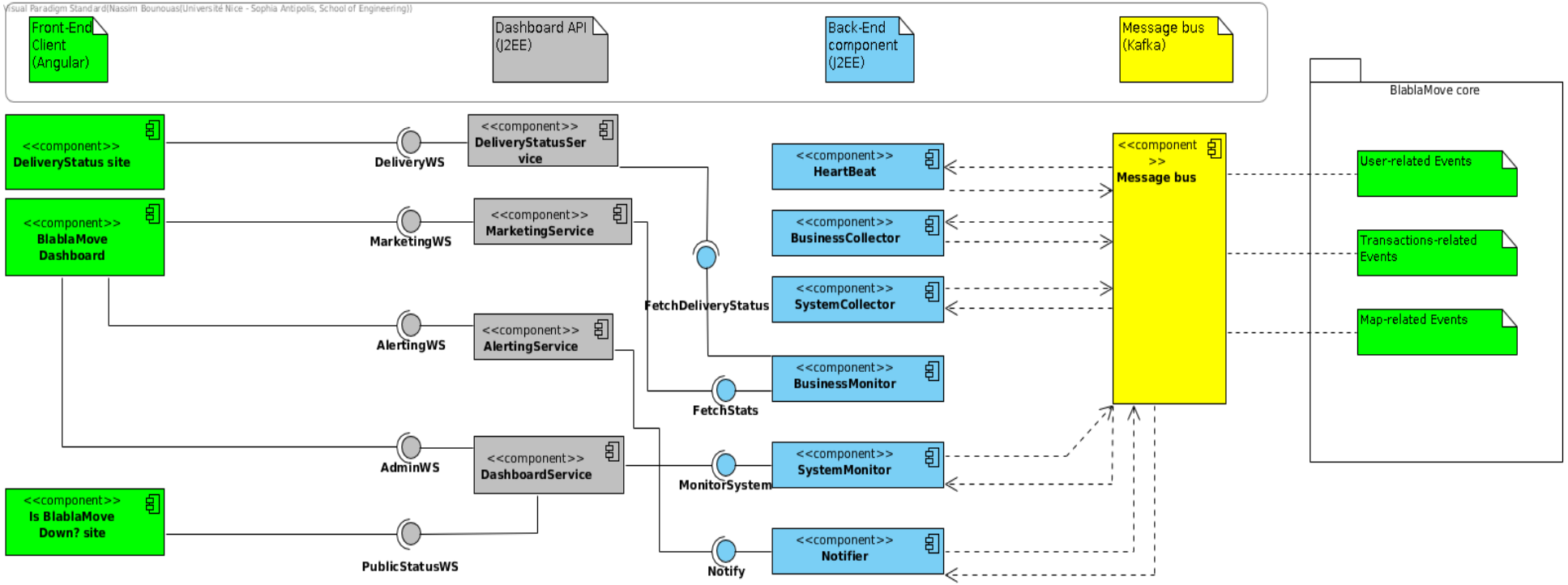
Notes

Dashboard

https://play.grafana.org/d/000000012/grafana-play-home?orgId=1
https://play.grafana.org/d/000000045/big-dashboard2?orgId=1
http://akveo.com/ngx-admin/#/pages/iot-dashboard

Week 42 (Rendu Mercredi 17 Octobre)

Lors de la séance de Vendredi nous avons voulu démarrer l'implémentation de notre système. Nous nous sommes très rapidement rendu compte que notre architecture n'était pas viable et avons préféré la revoir plutôt que de continuer tout en sachant que nos efforts seraient vains.


Visual Paradigm Standard(Nassim Bounouas(Université Nice - Sophia Antipolis, School of Engineering))
Front-End Client (Angular)
Dashboard API (J2EE)
Back-End component (J2EE)
Message bus (Kafka)
<<component>> DeliveryStatus site
<<component>> BlablaMove Dashboard
<<component>> Is BlablaMove Down? site
DeliveryWS
MarketingWS
AlertingWS
AdminWS
PublicStatusWS
<<component>> DeliveryStatusService
<<component>> MarketingService
<<component>> AlertingService
<<component>> DashboardService
FetchDeliveryStatus
FetchStats
MonitorSystem
Notify
<<component>> HeartBeat
<<component>> BusinessCollector
<<component>> SystemCollector
<<component>> BusinessMonitor
<<component>> SystemMonitor
<<component>> Notifier
<<component>> Message bus
BlablaMove core
User-related Events
Transactions-related Events
Map-related Events

Nos interfaces :

MarketingWS :

List<City> GetActiveCities();
List<Transaction> GetTransactByCity(City c, Date from, Date to);
LinkedList<CityReport> GetMostActiveCitiesAllTime();

AdminWS :

ConnectionLog GetLast24hConnections();
TransactLog GetLast24hTransactions();

PublicStatusWS :

float GetAvailabilityRate();
Date GetLastHeartbeatDate();
List<Tuple<Date, Incident>> GetLast24hIncidents();
void ReportAPlatformProblem(IncidentReport report);

DeliveryWS/AlertingWS

Ces interfaces seront décrites plus tard dans le projet, elles ne sont pas la priorité.

Notre présentation du mois de novembre fera usage des interfaces suivantes :

MarketingWS :
LinkedList<CityReport> GetMostActiveCitiesAllTime();

AdminWS :
ConnectionLog GetLast24hConnections();

StatusClientWS :
Date GetLastAlive();
List<Tuple<Date, Incident>> GetLast24hIncidents();
void ReportAPlatformIncident(IncidentReport report);

De cette manière nous pourrons présenter nos deux interfaces utilisateurs (Client et interne à Blablamove) répondant aux besoins des trois personas présentés la semaine dernière.

Les scénarios correspondront donc à :

- Hugo (Marketing) consulte une carte des chaleurs représentant les villes les plus actives depuis le début de la plateforme
- Mathieu (Admin Sys) consulte un rapport lui indiquant les connexions à la plateforme sur les 24 dernières heures. Il peut ainsi détecter et analyser les pics de charge.
- Alex (Client) remarque un ralentissement de la plateforme, il accède à l'interface permettant de consulter l'état des services et déclare un incident car il n'arrive pas à utiliser Blablamove.