



Architectures



[Composants web : une architecture web dans le cadre du portail Masanté.be](#)

Architectures des services de base de la plate-forme eHealth

1. [Introduction](#)
2. [Développement d'un projet dans le cadre de la santé en ligne : Ce qu'il faut prévoir, comprendre et définir](#)
 1. [Contraintes en matière d'identification et de gestion des accès](#)
 1. [Enregistrement](#)
 2. [Authentification](#)
 3. [Autorisation](#)
 2. [Contraintes en matière d'identification et de sécurité de l'information](#)
 1. [Confidentialité](#)
 2. [Intégrité](#)
 3. [Définition des standards de communication \(langages/protocoles\)](#)
 4. [Définition d'un ou plusieurs types de flux](#)
 1. [Volet 'identification et gestion des Accès' > distinction entre](#)
 1. [une application destinée à fonctionner sur le dispositif mobile de l'utilisateur \(native app/public client\)](#)
 2. [une application 'server based, hébergée par un partenaire et 'appelée' par l'utilisateur pour utilisation sur son outil mobile \(confidential client\)](#)



3. une application ne nécessitant pas d'intervention humaine, destinée à fonctionner automatiquement de serveur à serveur, pour la mise à jour automatique de banques de données par exemple (system client)
2. Volet 'Sécurité de l'information' > plusieurs aspects
3. Cas pratiques schématisés
 1. Enregistrement d'une clé publique (use case : enregistrement d'une clé dans le cadre de la demande d'un certificat eHealth au sein d'une architecture de type SOAP)
 2. Enregistrement d'une clé symétrique (use case : enregistrement d'une clé dans le cadre de Recip-e)
 3. Destinataire connu, communication synchrone (use case le plus fréquent : lorsqu'un client doit contacter directement un service de plate-forme eHealth qui impose le système d'encryption)
 4. Destinataire connu, communication asynchrone (use case: eHealthBox)
 5. Destinataire inconnu (use case : Recip-e)

1. Introduction

Dans le cadre du développement et de la maintenance de ses projets et services, la plate-forme eHealth propose différentes structures et organisations des systèmes informatiques, appelés « architectures ».

Ces modèles sont élaborés sur base des besoins des partenaires mais sont tenus de respecter certaines normes de qualité et de sécurité. Ils sont soumis à de constantes évolutions, en relation directe avec le secteur.

Lors du démarrage d'un projet, il importe donc de comprendre les différents systèmes proposés afin d'assurer une mise en place optimale des différents composants mais également d'anticiper les évolutions futures possibles.

La plate-forme eHealth propose principalement 2 types d'architectures :

- une architecture de type SOAP (Simple Object Access Protocol), destinée aux applications et services dont l'objectif est de fonctionner sur un seul dispositif, un seul ordinateur ;
- une architecture de type REST (Representational State Transfert), destinée aux applications et services dont l'objectif est de fonctionner sur plusieurs dispositifs (simultanément un ordinateur, un smartphone, une tablette...).



Comme mentionné préalablement, l'informatique est un domaine en constante évolution. Au démarrage de la plate-forme eHealth, l'utilisation de dispositifs mobiles tels que les tablettes et smartphones n'en était qu'à ses débuts. Raison pour laquelle l'architecture de type SOAP a majoritairement été développée et structure aujourd'hui encore de nombreux systèmes mis en place avec nos partenaires. La maintenance et le support pour ce modèle demeurent aujourd'hui parmi nos missions et responsabilités. Néanmoins, parce qu'il n'est pas recommandé pour le développement de projets de type mobiles (il ne permet notamment pas l'encryption des messages), la priorité est logiquement donnée à la promotion de l'architecture de type REST.

2. Développement d'un projet dans le cadre de la santé en ligne : ce qu'il faut prévoir, comprendre et définir

2.1. Le projet doit intégrer des contraintes en matière d'identification et de gestion des accès

Afin de permettre l'accès mobile aux services eHealth, nous devons être en mesure d'authentifier TOUS les utilisateurs qui ont besoin d'utiliser les services de la plate-forme eHealth, quel que soit l'appareil ou le système utilisé pour se connecter.

Dans l'ensemble, nous distinguons deux catégories d'utilisateurs de nos services :

- les personnes (citoyens belges ou étrangers, professionnels, membres d'une organisation, mandataires) ;
- les systèmes.

Il doit être possible de construire une identité numérique pour chacun d'entre eux.

2.1.1. Enregistrement

Tous les utilisateurs doivent être enregistrés dans une source authentique accessible à la plate-forme eHealth (directement ou indirectement) :

- les personnes présentes dans le Registre national avec un NISS pour citoyens belges ou NISS BIS pour étrangers (les groupes cibles de la plate-forme eHealth incluent les citoyens belges et les étrangers qui vivent ici ou à l'étranger) ;
- les systèmes doivent appartenir à une organisation qui peut être identifiée de manière univoque dans une source authentique pour le type spécifique d'organisation.

Tout utilisateur doit être en mesure de prouver son identité en ligne avec une clé numérique. Au moins une clé doit lui être remise lors de l'enregistrement.



2.1.2. Authentification

L'authentification doit être supportée pour tous les types de clients : web (navigateur), natif (application mobile), desktop, serveur (backend, batch).

Pour s'authentifier, l'utilisateur doit utiliser l'une de ses clés numériques pour prouver qu'il est bien celui qu'il prétend être. Le modèle d'identité fédérée de la plate-forme eHealth doit être réutilisable pour tous les utilisateurs.

Toutes les clés numériques doivent répondre à des exigences minimales de sécurité.

Une personne doit pouvoir utiliser plusieurs dispositifs pour s'authentifier vis-à-vis de nos services.

Une personne doit être en mesure de choisir un profil d'utilisateur applicable (c.-à-d. citoyen, qualité, appartenance à une organisation, mandat) qui sera utilisé pour l'authentification vis-à-vis de nos services.

Il doit être possible de transmettre l'identité choisie aux ressources demandées ou celles-ci doivent pouvoir la récupérer.

2.1.3. Autorisation

Les autorisations doivent être basées sur l'identité numérique choisie pour chacune des ressources demandées.

Il doit être possible de propager les autorisations aux ressources demandées ou celles-ci doivent pouvoir les obtenir.

Il doit être possible de laisser l'utilisateur décider s'il souhaite ou non donner des autorisations à l'application cliente qui utilisera ces autorisations en son nom.

Les utilisateurs doivent pouvoir révoquer les autorisations accordées.

2.2. Le projet doit intégrer des contraintes en matière d'identification et de sécurité de l'information

2.2.1. Confidentialité

Toute communication entre le client et le serveur doit être considérée comme confidentielle et doit être protégée contre toute interception, au moins lorsqu'elle transite par un support non sécurisé comme Internet.



Les données médicales doivent être protégées au niveau du message afin d'empêcher la divulgation des données lorsqu'on passe d'un point à un autre sur le réseau. Si le cryptage de bout en bout entre l'expéditeur d'origine et le destinataire final n'est pas nécessaire, il doit au moins être configuré point à point entre ces deux parties de sorte que les données médicales ne soient jamais envoyées sans protection entre deux points. La question de savoir si le point à point est suffisant est à décider par projet.

Les utilisateurs doivent pouvoir signer et chiffrer des messages sur différents appareils (ordinateur portable, smartphone et tablette) sans devoir transférer et exposer des clés numériques entre ces appareils.

2.2.2. Intégrité

Lorsque des données médicales sont envoyées du client au serveur, elles doivent être signées au niveau du message pour assurer l'intégrité du contenu.

2.3. Le projet doit définir les standards de communication parmi ceux proposés

Identity & Access Management

Voici la liste des langages/protocoles proposés :

- [SAML 2.0](#)
- [Oauth 2.0](#)
- [OIDC 1.0](#)
- [JWT](#)
- [Signed JWT Assertion](#)
- [PKCE](#)

Information Security

Voici la liste des langages/protocoles proposés :

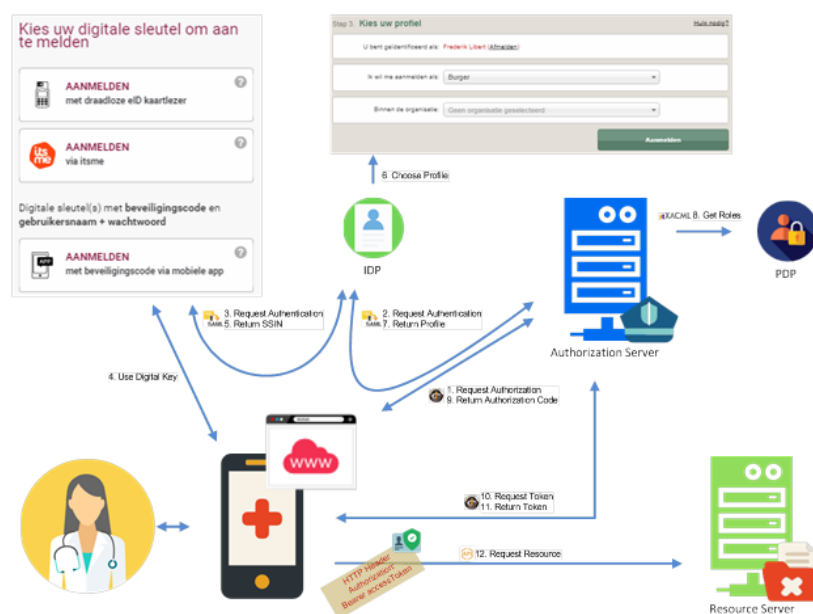
- [TLS](#)
- [JWS](#)
- [JWE](#)
- [JWK](#)
- [WebAuthn](#)



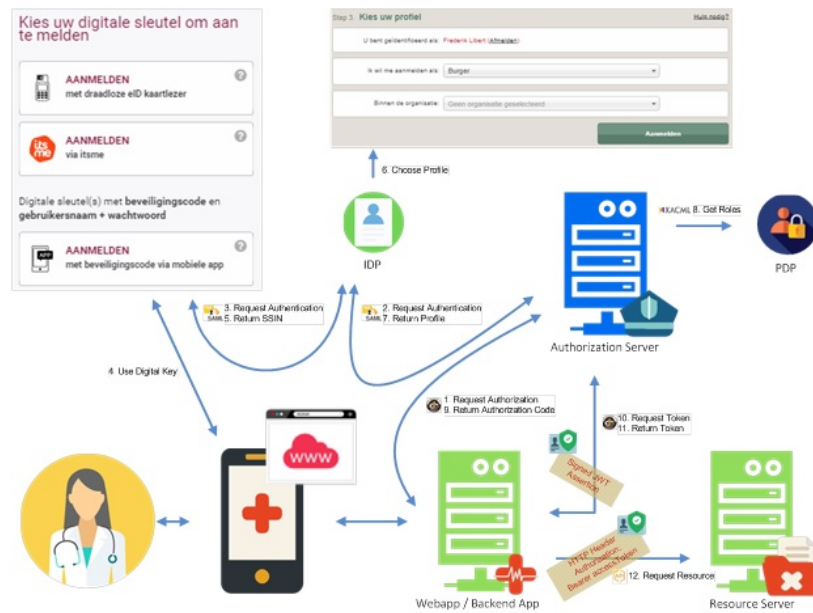
2.4. Le projet doit définir un ou plusieurs types de flux parmi ceux proposés

2.4.1. En ce qui concerne le volet « identification et gestion des Accès », il y a lieu de faire la distinction entre ces applications

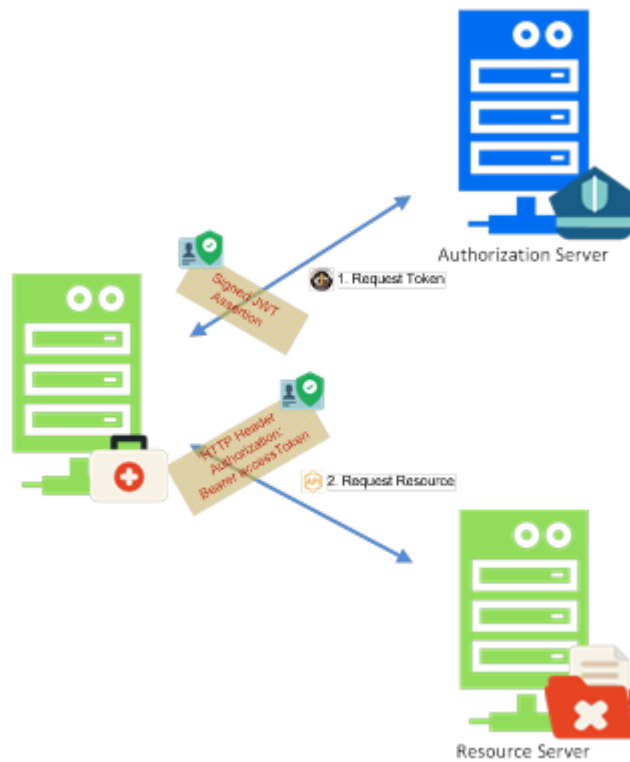
2.4.1.1. Une application destinée à fonctionner sur le dispositif mobile de l'utilisateur (native app/public client)



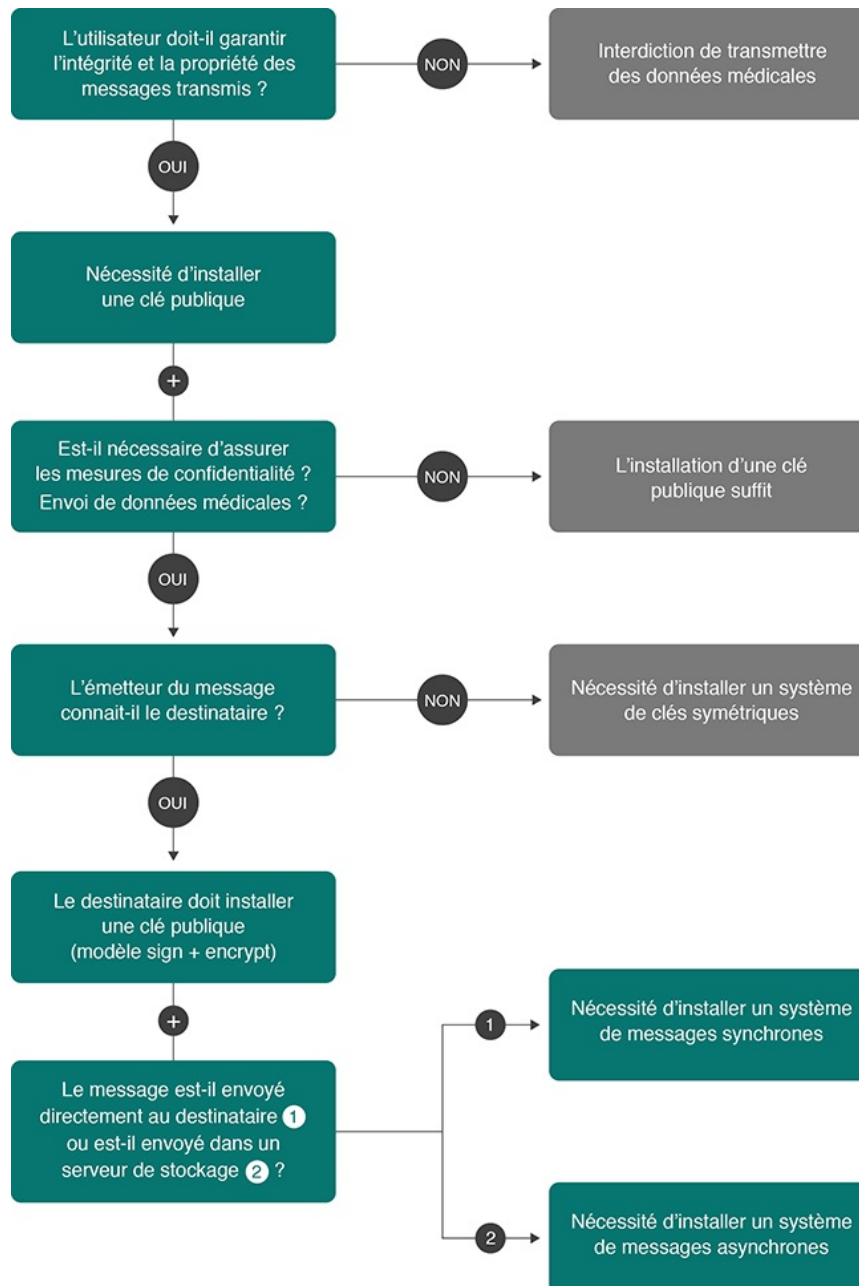
2.4.1.2. Une application « server based », hébergée par un partenaire et « appelée » par l'utilisateur pour utilisation sur son outil mobile (confidential client)



2.4.1.3. Une application ne nécessitant pas d'intervention humaine, destinée à fonctionner automatiquement de serveur à serveur, pour la mise à jour automatique de banques de données par exemple (system client)

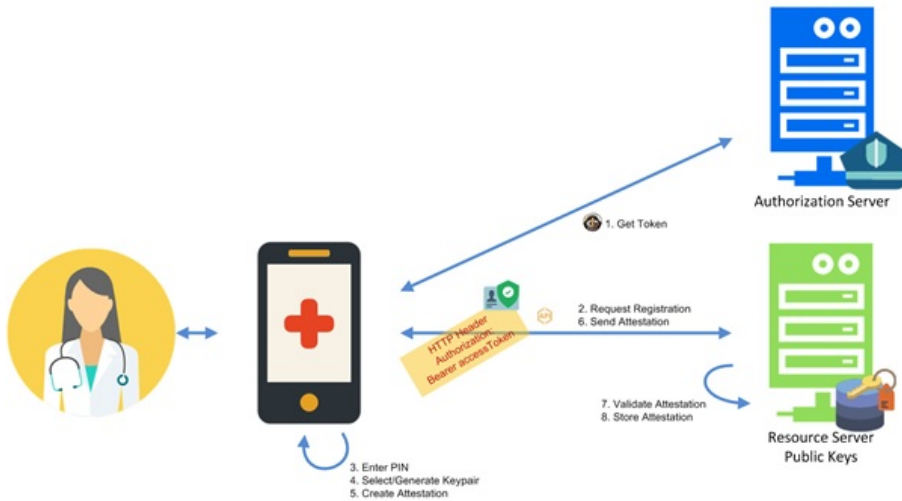


2.4.2. En ce qui concerne le volet « Sécurité de l'information », il y a lieu de s'interroger sur plusieurs aspects

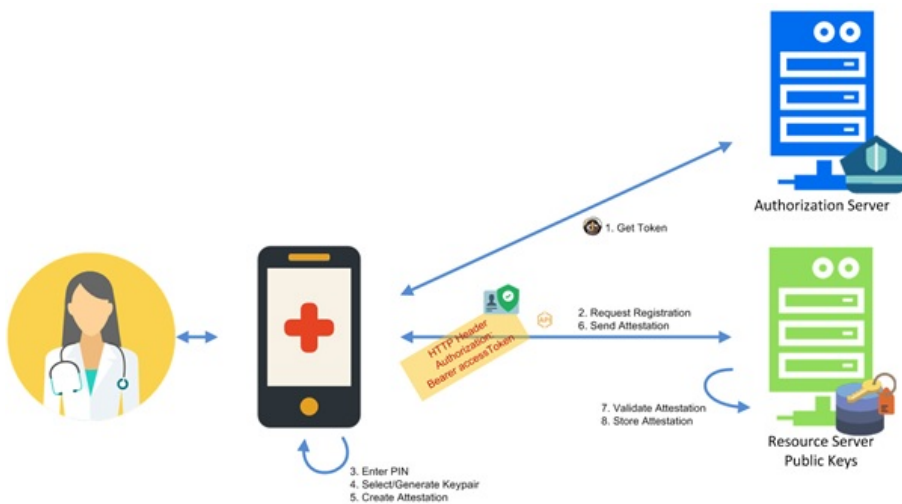


3. Cas pratiques schématisés

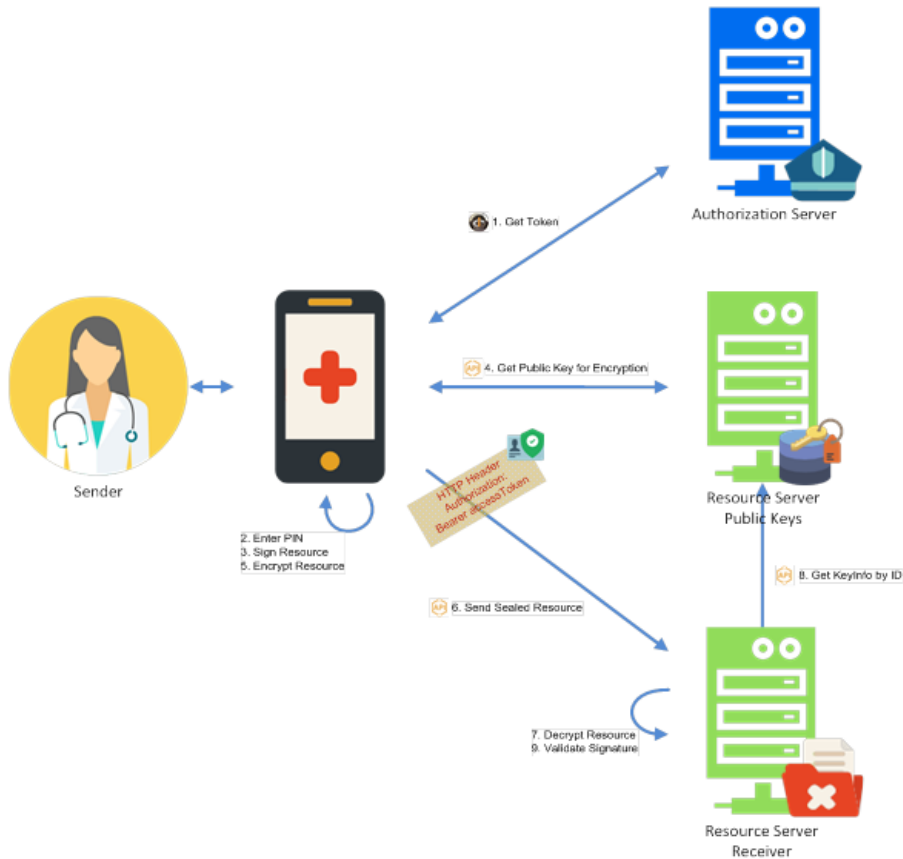
3.1. Enregistrement d'une clé publique (use case : enregistrement d'une clé dans le cadre de la demande d'un certificat eHealth au sein d'une architecture de type SOAP)



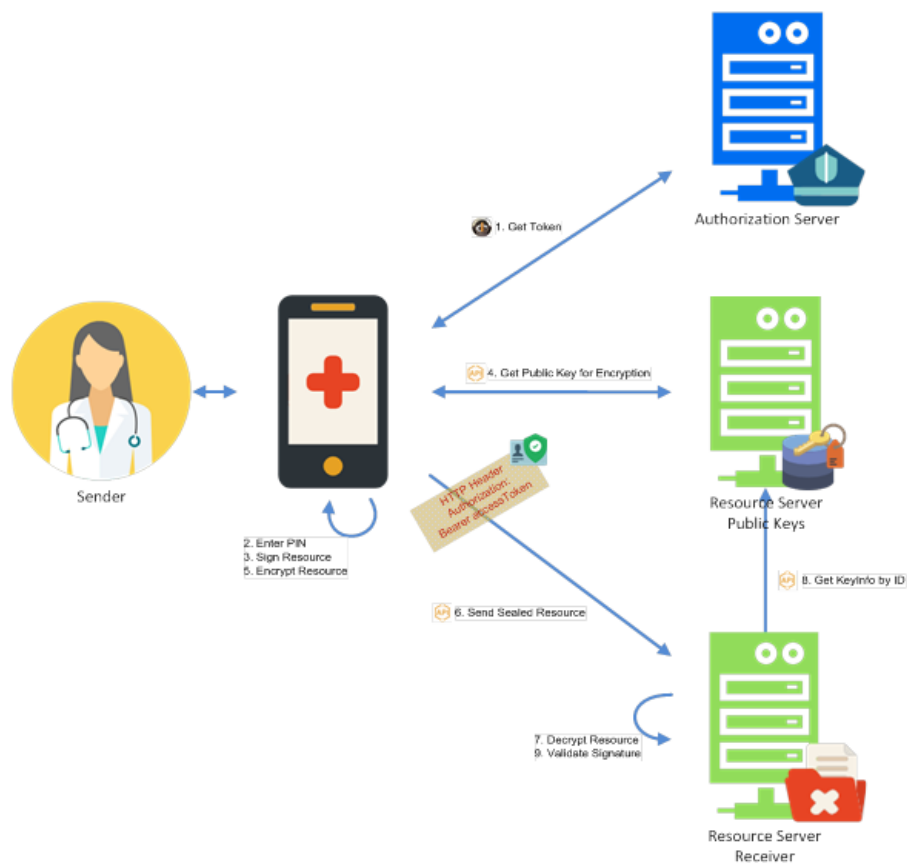
3.2. Enregistrement d'une clé symétrique (use case : enregistrement d'une clé dans le cadre de Recip-e)



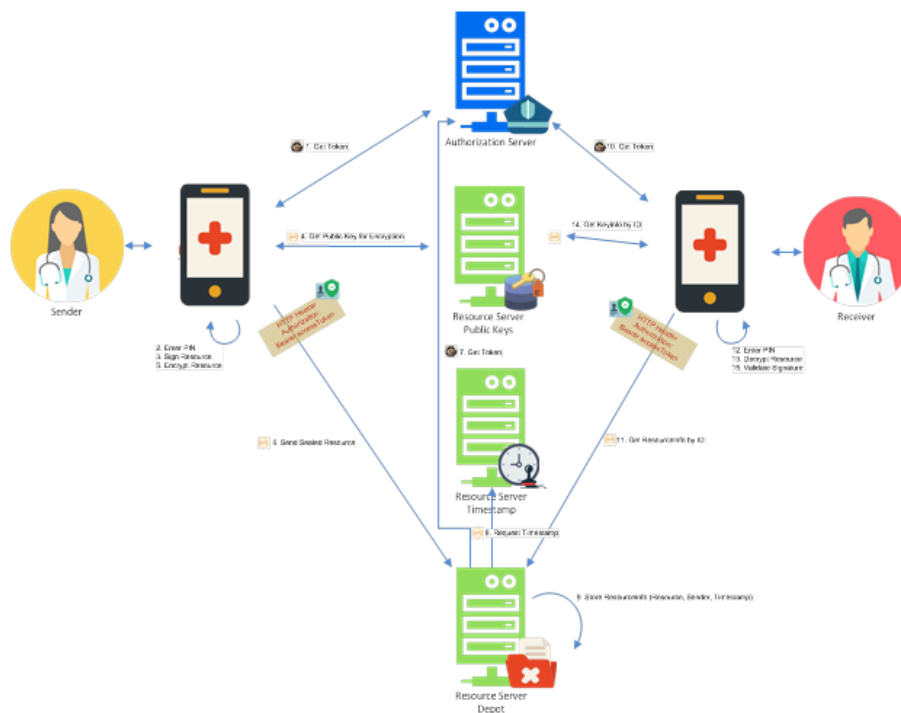
3.3. Destinataire connu, communication synchrone (use case le plus fréquent : lorsqu'un client doit contacter directement un service de plate-forme eHealth qui impose le système d'encryption)



3.4. Destinataire connu, communication asynchrone (use case : eHealthBox)



3.5. Destinataire inconnu (use case : Recip-e)



Composants web : une architecture web dans le cadre du portail Masanté.be

L'architecture du portail Masanté.be est de plus en plus basée sur l'utilisation de composants web. Cette architecture web offre, à l'instar d'une API, la possibilité de donner accès à des sources authentiques externes au moyen d'une application web. Les composants web offrent l'avantage supplémentaire que le fournisseur du composant (qui est typiquement le propriétaire de la source authentique de données) peut définir de quelle manière les données sont présentées dans l'application web qui intègre le composant. Ceci réduit fortement le risque d'erreurs dans l'affichage des données et réduit aussi les frais de développement pour les gestionnaires des applications web qui intègrent le composant (la couche de présentation est en effet fournie avec le composant).

Une introduction générale au sujet des composants web est disponible via le lien suivant :

[Introduction - webcomponents.org](https://www.webcomponents.org)

Dans le cadre du portail Masanté.be, Smals propose un [guide technique via le GitHub pour les parties souhaitant proposer des composants web via le portail ou souhaitant intégrer les composants existants du portail dans leur propre application web \(en anglais\)](#).

