Descrizione testuale (soluz. 1)

La soluzione adottata per implementare le nuove funzionalità prevede l'introduzione di componenti di tipo *make*. Trattandosi di funzionalità innovative e strettamente legate al sistema di legacy non è possibile affidarsi a software COTS già esistenti. Diventa quindi necessario adottare un approccio make capace di soddisfare tutte le esigenze di WWNS.

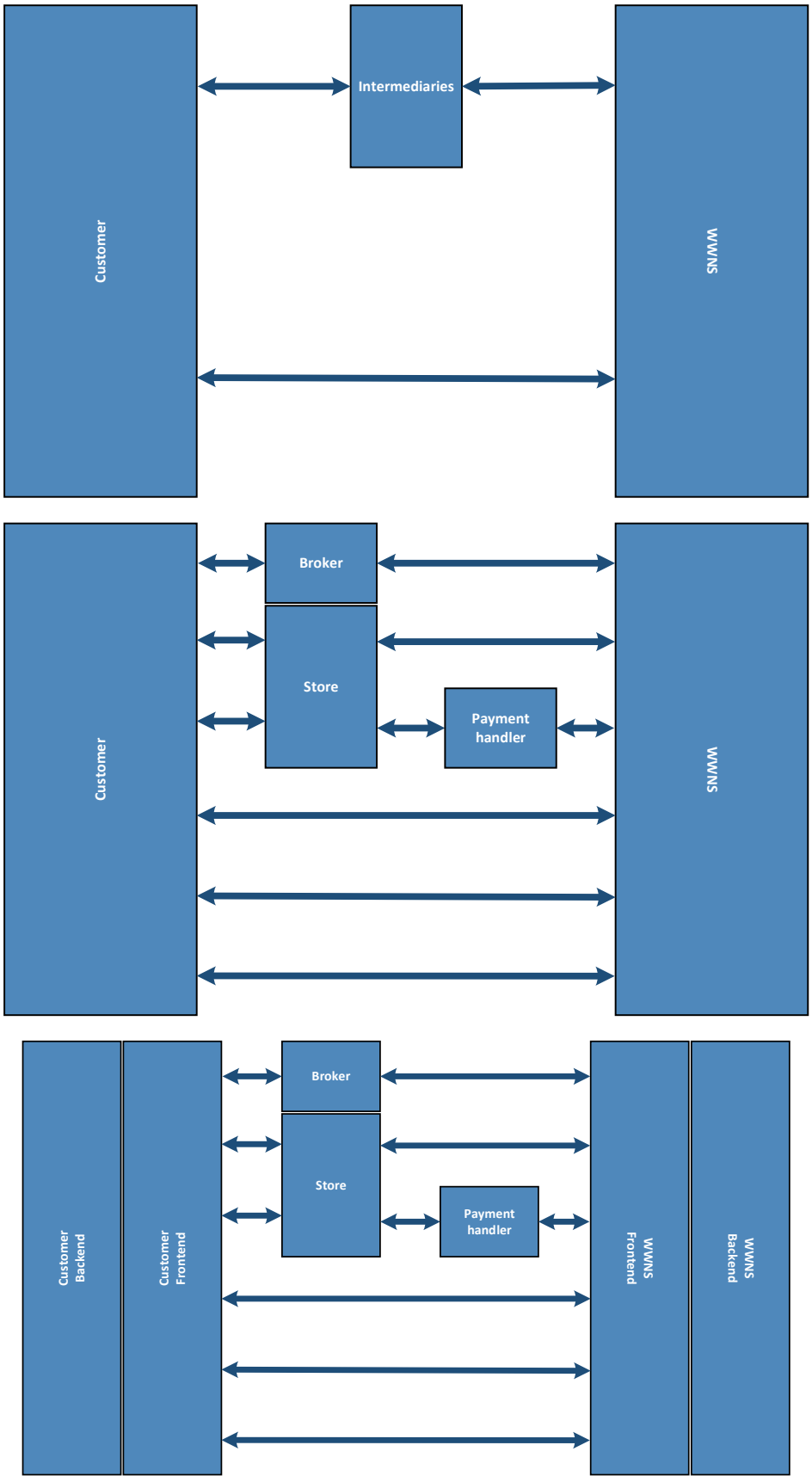
La gestione delle nuove funzionalità è *on premise* per motivi di maggiore controllo e integrazione con le funzionalità già esistenti.

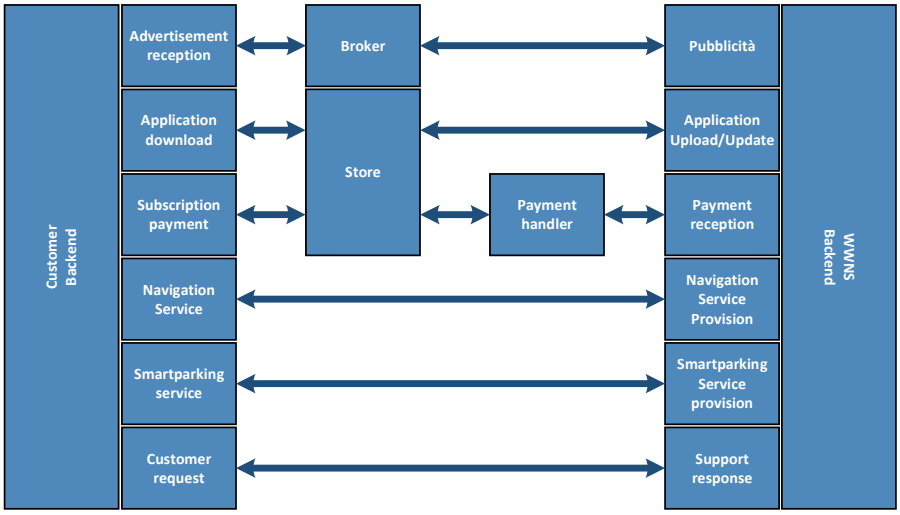
Questa soluzione prevede la sostituzione di uno dei moduli del party level, navigator, con un modulo più ampio chiamato "core navigation system" che lo ingloba ed estende le sue funzionalità.

Il funzionamento del nuovo modulo non prevede l'aggiunta di nessuna nuova base di dati, ma sfrutta quelle già esistenti per offrire i nuovi servizi.

Si tratta di una soluzione "economica" per implementare le nuove funzionalità, sia per numero di componenti aggiunti, sia perché non richiede l'introduzione di nessuna nuova base di dati. Tuttavia questo potrebbe ridurre la qualità dei servizi offerti, producendo più frequentemente (rispetto la soluzione 2) risultati con probabilità di validità incerte.

Aspetto organizzativo (soluz. 1) – Diagrammi Organizzativi fino al livello 4





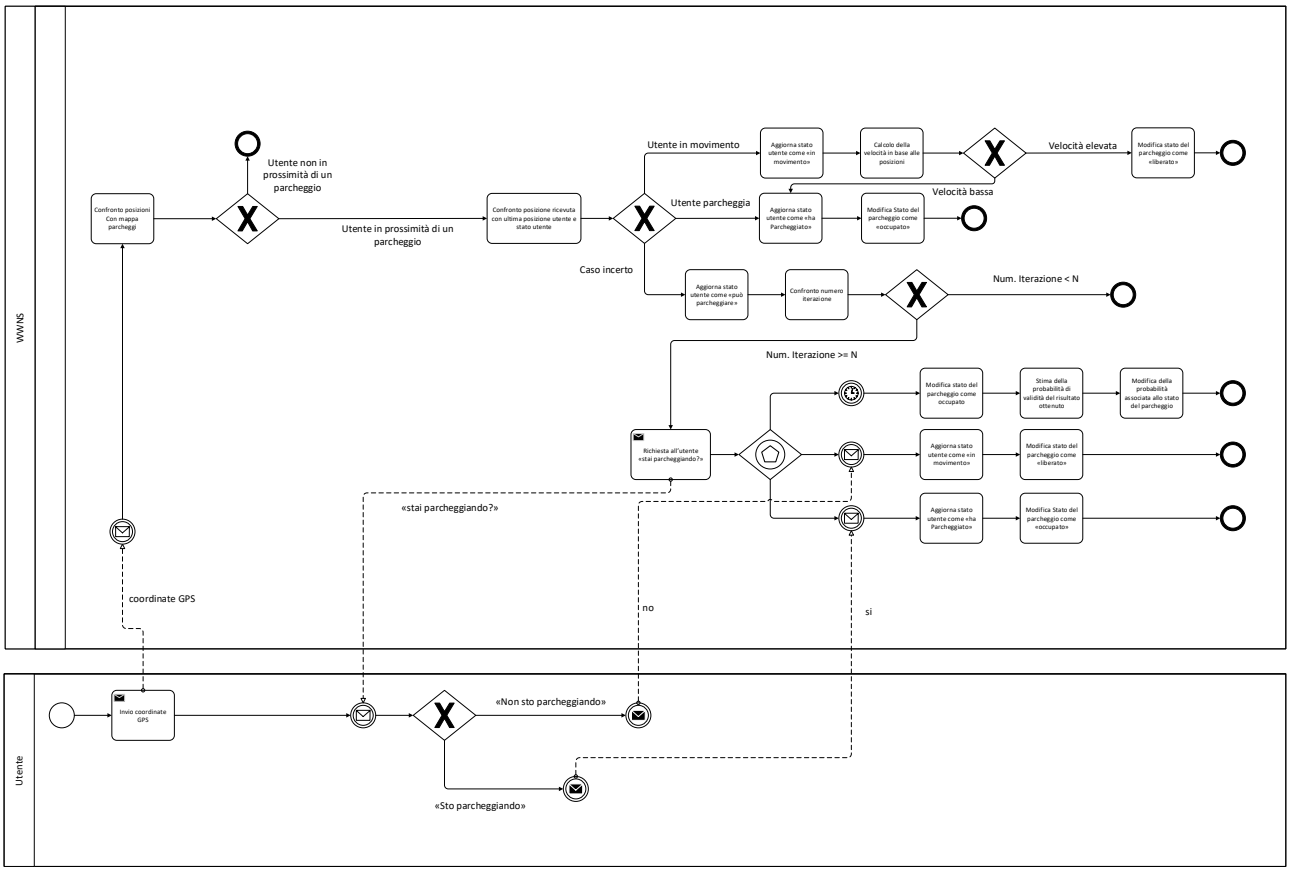
Componenti di front-end di WWNS:

- adversting, si occupa di fornire contenuti pubblicitari al broker;
- application upload/update, carica il codice e aggiornamenti dell'applicazione sui sistemi dello store;
- payment reception, riceve i pagamenti degli abbonamenti tramite il payment handler;
- navigation service, dialoga con il sistema di back–end per fornire i servizi di navigazione agli utenti;
- smartparking service, dialoga con il sistema di back–end per fornire il servizio di SmartParking agli utenti;
- support response, modulo di front-end che fornisce il supporto ai clienti.

Intermediari:

- Broker, fornisce i contenuti pubblicitari di WWNS ai potenziali clienti.
- Store, immagazzina il codice di WWNavigApp e relativi aggiornamenti forniti da WWNS per fornirli in modo asincrono al cliente. Media anche il pagamento di abbonamenti tra cliente e payment handler.
- Payment handler, media il pagamento tra cliente e WWNS tramite l'interfaccia dello store.

BPMN processo:
valutazione dello stato dei parcheggi



Questo processo è iterato più volte e periodicamente, ogni volta che un utente invia la propria posizione a WWNS. Ogni iterazione sulla base dei dati ricevuti e di quelli precedentemente memorizzati aggiorna stato utente, stato dei parcheggi e posizione dell'utente.

Il blocco "confronto posizione ricevuta con ultima posizione utente e stato utente" cerca di dedurre sulla base dei dati forniti il comportamento dell'utente.

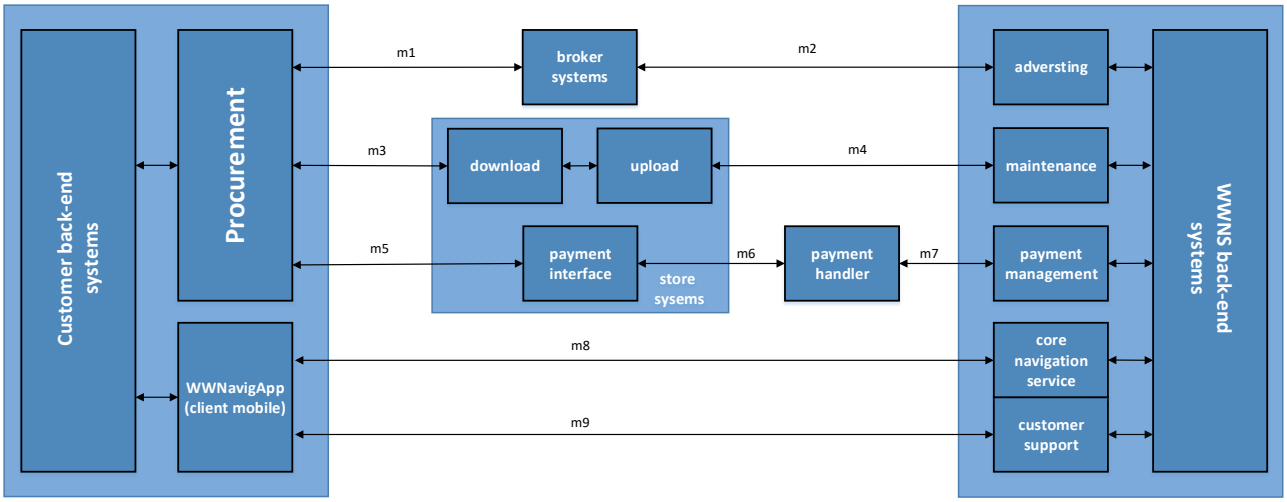
Supponiamo che un utente dopo aver parcheggiato porti con sé il cellulare, quindi l'unico modo che ha CPID per capire se un utente abbia effettivamente parcheggiato è osservando un cambiamento di velocità di movimento dell'utente. La velocità è calcolata tramite confronto tra ultima posizione utente, la posizione ricevuta e l'intervallo di tempo tra le due rilevazioni.

Le coordinate GPS che l'utente invia a WWNS sono associate all'istante di tempo in cui sono state rilevate e ad uno user ID.

Se un utente permane in uno stato incerto ("può parcheggiare") dopo N iterazioni di questo processo, WWNS chiede all'utente le sue intenzioni.

Architettura funzionale (soluz. 1)

Architettura market level



Message set	Content Exchanged
m1	Customer search request, provider search result
m2	Advertising content provision
m3	Download app, downloading notification
m4	Upload/update app
m5	Payment data
m6	Payment order
m7	Payment notification
m8	User location, navigation instructions
m9	Customer message, support response

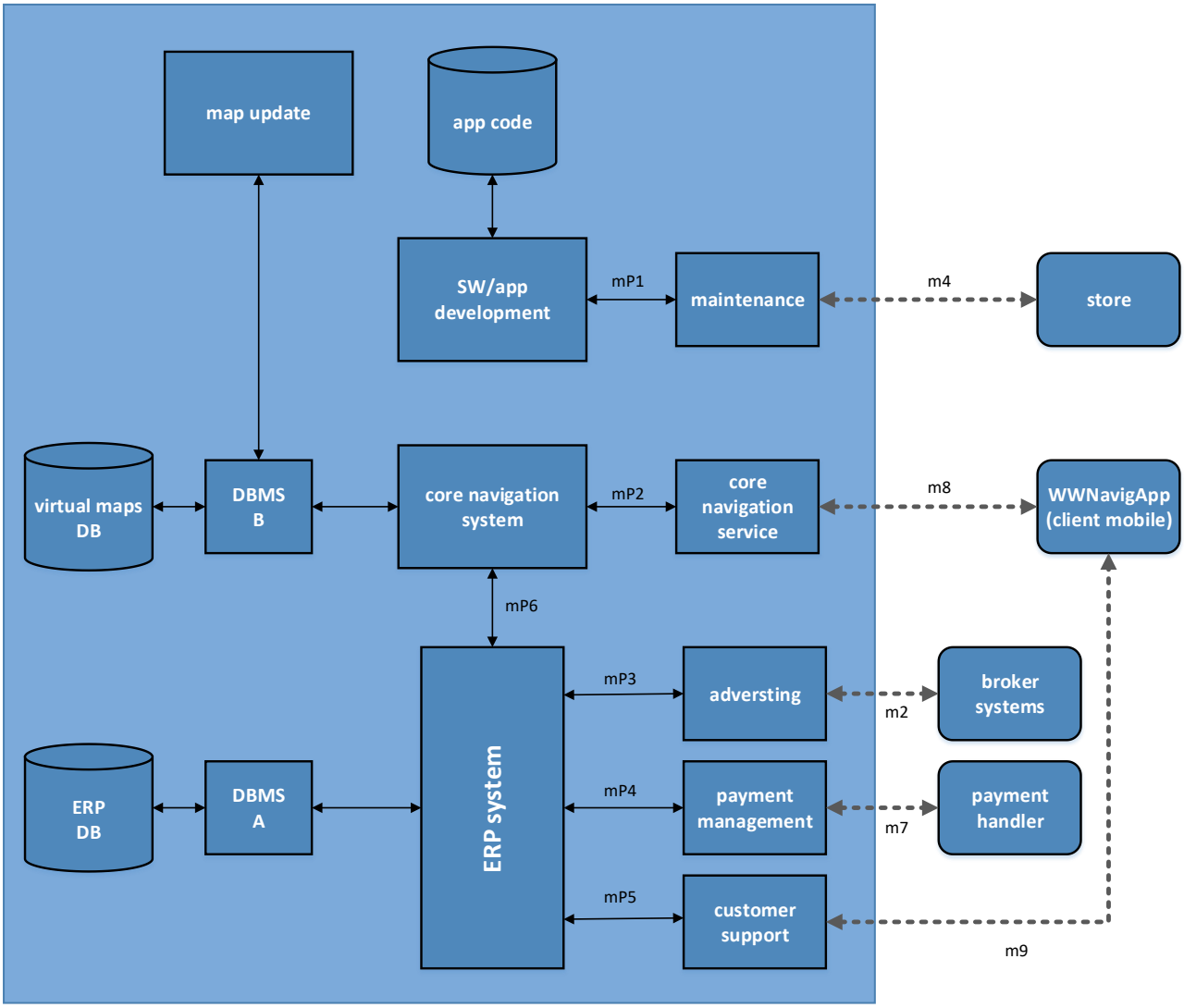
Componenti di front-end di WWNS:

- adversting, si occupa di fornire contenuti pubblicitari al broker;
- maintenance, carica il codice e aggiornamenti dell'applicazione sui sistemi dello store;
- payment management, gestisce i pagamenti degli abbonamenti tramite il payment handler;
- core navigation service, dialoga con il sistema di back–end per fornire i servizi di navigazione e SmartParking agli utenti;
- customer support, modulo di front-end che fornisce il supporto ai clienti.

Tipi di canali di comunicazione:

- rete internet, tutte i canali di comunicazione rappresentati poggiano sui servizi internet.

Architettura party level



Message set	Content Exchanged
mP1	Invio applicazione e relativi aggiornamenti
mP2	In ingresso a core navigation system arrivano dati del tipo: posizione, istante di tempo in cui la posizione è stata rilevata, nome utente… In uscita vengono inviati grafi (rappresentanti porzioni delle mappe virtuali) contenenti i percorsi verso le destinazioni richieste dagli utenti.
mP3	Scambio di contenuti pubblicitari
mP4	Notifica dei pagamenti nei sistemi di contabilità
mP5	Scambio di informazioni relativo al supporto clienti
mP6	Il core navigation system si assicura di offrire il servizio di SmartParking solo agli utenti che hanno pagato l'abbonamento.

ENTERPRISE ARCHITECTURE

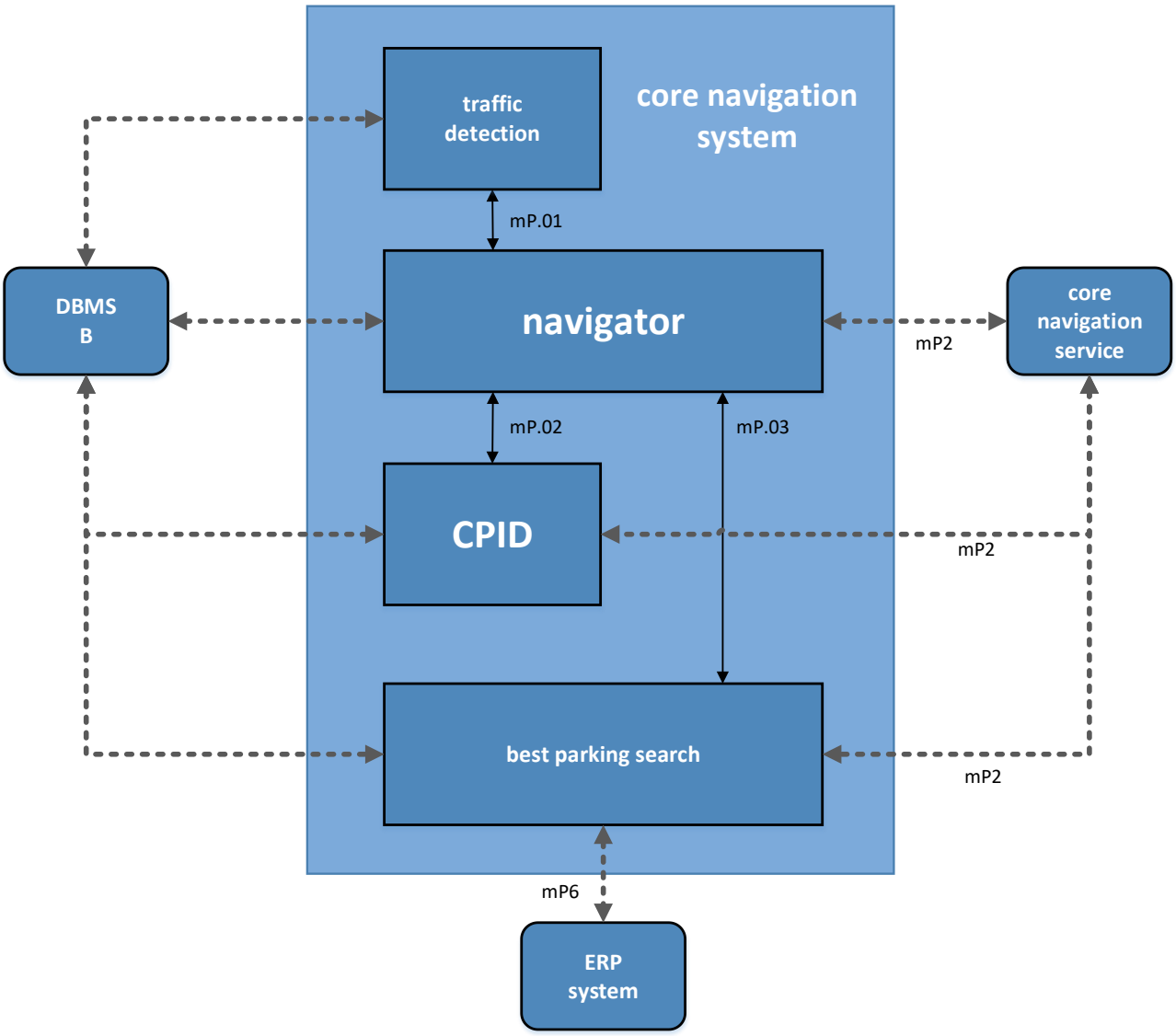
Componenti di back-end di WWNS:

- SW/app development, componente dedicato allo sviluppo di WWNavigApp e alla sua manutenzione tramite aggiornamenti.
- Core navigation system, componente che offre i servizi primari di WWNS, ovvero il servizio di legacy di navigazione e il servizio di SmartParking. Si occupa anche del rilevamento dello stato del traffico e dei parcheggi. Per offrire i sui servizi richiede periodicamente la posizione degli utenti.
- ERP system, insiemi di moduli software dedicati alla gestione della contabilità, dei contenuti pubblicitari e del servizio clienti.
- Map update, componente che si occupa di aggiornare (inserendo anche i parcheggi delle grandi aree metropolitane designate) le mappe virtuali su cui poggiano i servizi offerti dal core navigation system.

Basi di dati e altre strutture di dati:

- ERP DB, DB che immagazzina i dati utilizzati dai moduli ERP.
- Virtual maps DB, DB che memorizza le mappe virtuali utilizzate dal core navigation system.
- App code, immagazzina il codice delle diverse versioni di WWNavigApp.

Architettura system level (soluz. 1)



Message set	Content Exchanged
mP.01	Inoltro delle posizioni rilevate da navigator a traffic detection
mP.02	Inoltro delle posizioni rilevate da navigator a CPID
mP.03	Invio delle destinazioni relative al *miglior parcheggio* trovato da best parking search a navigator

Componenti del core navigation system:

- Navigator, componente di legacy che può ricevere due tipi di richieste:
    1. le destinazioni richieste dagli utenti e le loro *posizioni*[1] al momento della richiesta;
    2. oppure, periodicamente, solo la posizione degli utenti che fanno uso dell'app.

  In risposta al primo tipo di richiesta, grazie ai dati ricevuti, e tramite le mappe virtuali al quale accede attraverso il DBMS B, calcola il *miglior percorso*[2] verso la destinazione richiesta. Una volta trovato il percorso migliore lo invia all'utente tramite il componente di front-end core navigation service. In particolare invia una porzione di grafo/mappa virtuale all'utente che contiene il percorso richiesto, infatti si tratta dell'unica parte della mappa virtuale che in quel momento è di interesse all'utente. L'invio del percorso all'utente comprende anche una probabilità che si riferisce alla validità di quest'ultimo.

  In risposta al secondo tipo di richiesta invia all'utente una porzione del grafo circostante alla sua posizione, per offrirgli una visione parziale della mappa virtuale dell'area in cui si trova.

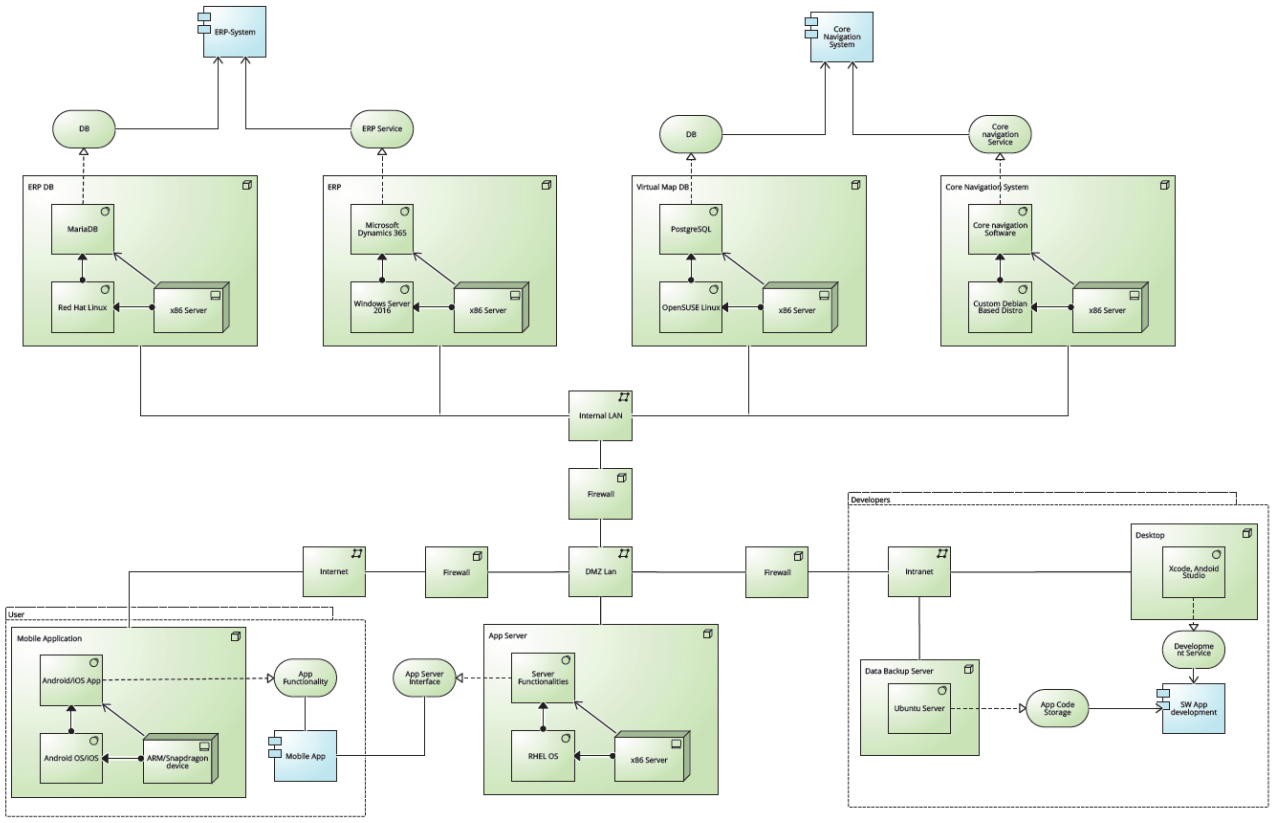
- Traffic detection, componente che implementa la funzionalità di rilevamento del traffico. Questo modulo riceve da navigator le posizioni rilevate periodicamente dagli utenti. Utilizza poi tali posizioni per monitorare lo stato del traffico nei nodi delle mappe virtuali (al quale accede tramite DBMS B), e se necessario li marca come hotspot, con conseguente aumento dei costi di percorrenza degli archi direttamente collegati.
- CPID (Customer Parking Intention Deduction), modulo software il cui ruolo è quello di capire le intenzioni degli utenti per rilevare lo stato dei parcheggi. Riceve le posizioni rilevate periodicamente da navigator, e le utilizza per capire se gli utenti sono in prossimità di un parcheggio, e se effettivamente hanno posteggiato; in tal caso il nodo speciale rappresentante il parcheggio sarà marcato come "occupato" all'interno della mappa virtuale. Nel caso in cui gli algoritmi di CPID non riescano ad ottenere dei risultati con una percentuale di errore accettabile chiedono conferma all'utente.
- Best parking search, su richiesta dell'utente questo modulo si occupa della ricerca del *miglior parcheggio*[3]. Chiede all'ERP di verificare che l'utente che richiede il servizio sia abbonato, e in caso positivo cerca il miglior parcheggio nella mappa virtuale e lo invia sotto forma di destinazione al navigator. Sarà poi il navigator a guidare l'utente verso la destinazione con i meccanismi di legacy già descritti.

I moduli CPID e best parking search permettono a WWNS di fornire il servizio di SmartParking.

È importante sottolineare che tutti i componenti citati non memorizzano le posizioni degli utenti in nessun set di dati corposo, ma le utilizzano per cambiare lo stato della mappa virtuale, per calcolare percorsi verso una destinazione, ecc. Dopo aver processato le posizioni queste vengono eliminate. Tuttavia, solo se necessario, è possibile tenere traccia delle ultime rilevate.

I componenti traffic detection e CPID associano le modifiche fatte alle mappe virtuali con delle probabilità che si riferiscono alla loro validità. Queste probabilità vengono poi prese in esame da navigator nei sui calcoli, per generare infine un valore di validità del risultato offerto all'utente.

Architettura fisica (soluz. 1)



Il diagramma in ArchiMate 3 rappresenta l'architettura fisica su cui si appoggia il Sistema Informativo relativo alla soluzione 1. In particolare si vuole descrivere l'architettura server su cui si appoggia il *Core navigation System*, rappresentato da un modulo apposito nel party level. Infatti in questa soluzione si suppone che l'intero pacchetto software di tipo make relativo a questa funzionalità possa operare su una singola macchina. Su questa macchina è installata una distribuzione creata *ad hoc* di Linux, basata su Red Hat. Inoltre, poiché si suppone che la funzionalità *core* di WWNS necessiti di grande capacità di calcolo si può presumere che esistano molteplici macchine che operino sulla stessa rete e svolgano le medesime funzioni.

Relativamente alla sezione delimitata dal grouping "developers" vale la considerazione che esistano molteplici macchine appartenenti agli sviluppatori (qui non descritte) che operino su una stessa rete LAN e condividano il codice sorgente dell'applicazione WWNavigApp. Questo è salvato su un server di backup naturalmente connesso alla stessa rete.

La rete della sezione *developing* e quella relativa alle funzionalità offerte dal servizio sono separate da firewall e da una zona demilitarizzata (su cui opera il server che si interfaccia con l'App, in quanto non hanno la necessità di comunicare in modo diretto.

Descrizione testuale (soluz.2)

La soluzione adottata per implementare le nuove funzionalità prevede l'introduzione di componenti di tipo *make*. Trattandosi di funzionalità innovative e strettamente legate al sistema di legacy non è possibile affidarsi a software COTS già esistenti. Diventa quindi necessario adottare un approccio make capace di soddisfare tutte le esigenze di WWNS.

La gestione delle nuove funzionalità è *on premise* per motivi di maggiore controllo e integrazione con le funzionalità già esistenti.

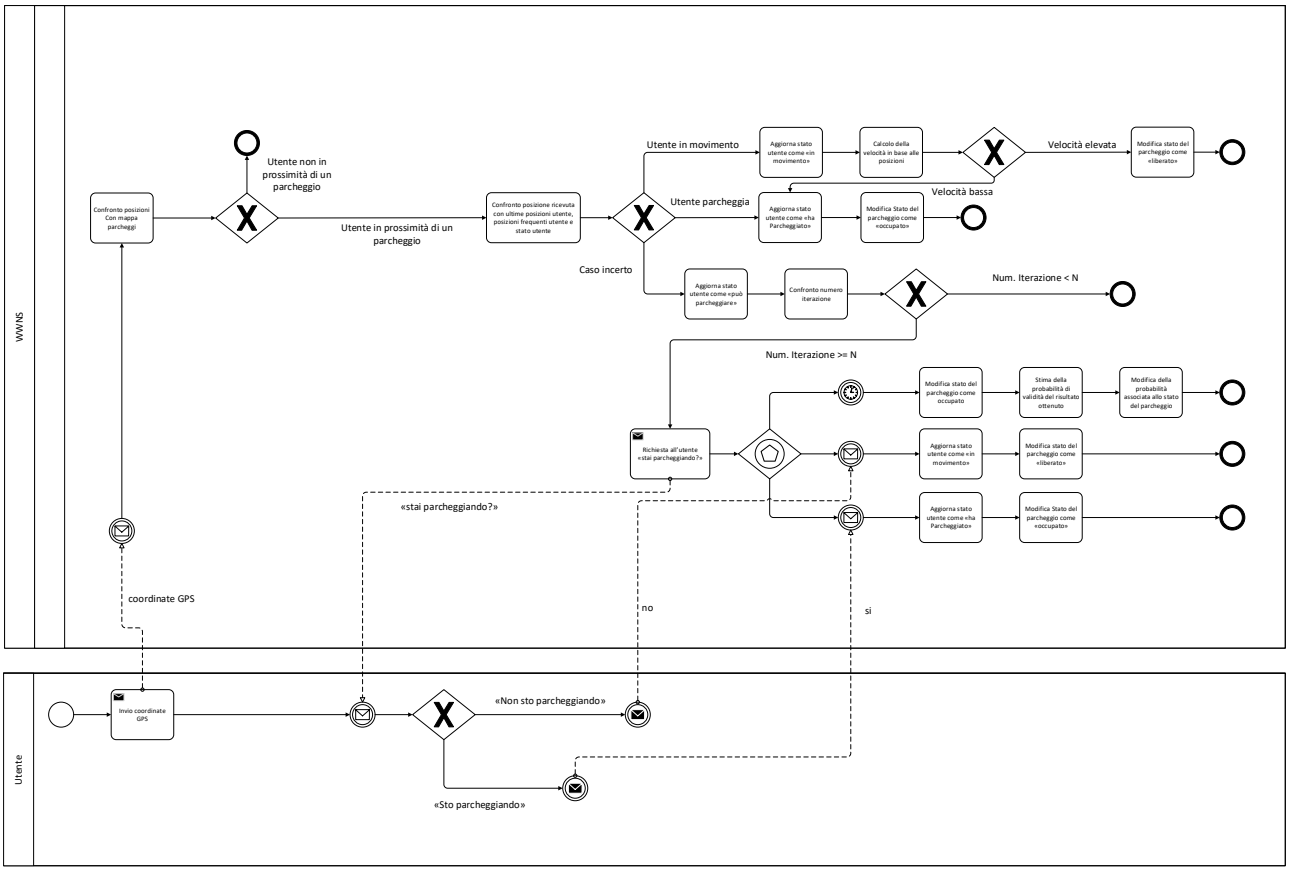
Questa soluzione prevede la sostituzione di uno dei moduli del party level, navigator, con un modulo più ampio chiamato "core navigation system" che lo ingloba ed estende le sue funzionalità.

Il funzionamento del nuovo modulo prevede l'aggiunta di una nuova base di dati il cui scopo è immagazzinare le posizioni recenti e frequenti degli utenti. Rispetto la soluzione 1 e necessario aggiungere un altro modulo (visibile a system level) che si occupa di immagazzinare e catalogare correttamente le posizioni in ingresso nel nuovo DB.

Si tratta della soluzione meno economica per implementare le nuove funzionalità, sia per numero di componenti aggiunti, sia perché richiede l'introduzione di una nuova base di dati. Il nuovo DB permette ai componenti adibiti alle nuove funzionalità di lavorare su una quantità di dati maggiori, e quindi di avere una visione più dettagliata dello stato del traffico e dei parcheggi. Il risultato è un servizio di qualità maggiore, che riduce la probabilità di ottenere risultati di dubbia validità.

Aspetto organizzativo (soluz. 2)

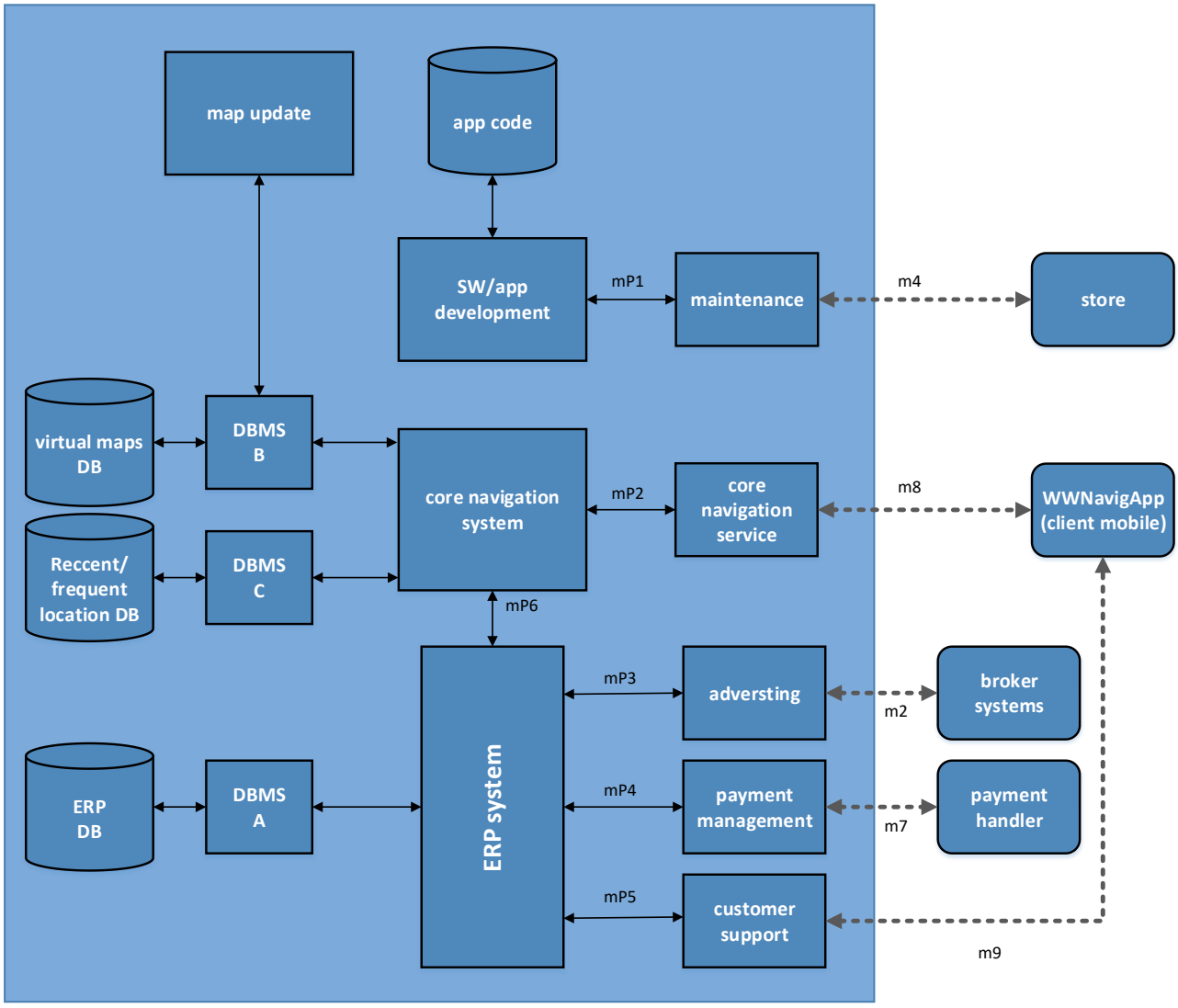
BPMN processo:
valutazione stato parcheggi



Il processo resta pressoché inalterato rispetto la variante della soluzione 1. La differenza sostanziale sta nel fatto che il blocco “Confronto posizione ricevuta con ultime posizioni utente, posizioni frequenti utente e stato utente” ha più dati su cui lavorare, permettendo a CPID di ottenere meno casi incerti rispetto la prima soluzione.

Architettura funzionale (soluz. 2)

Architettura party level

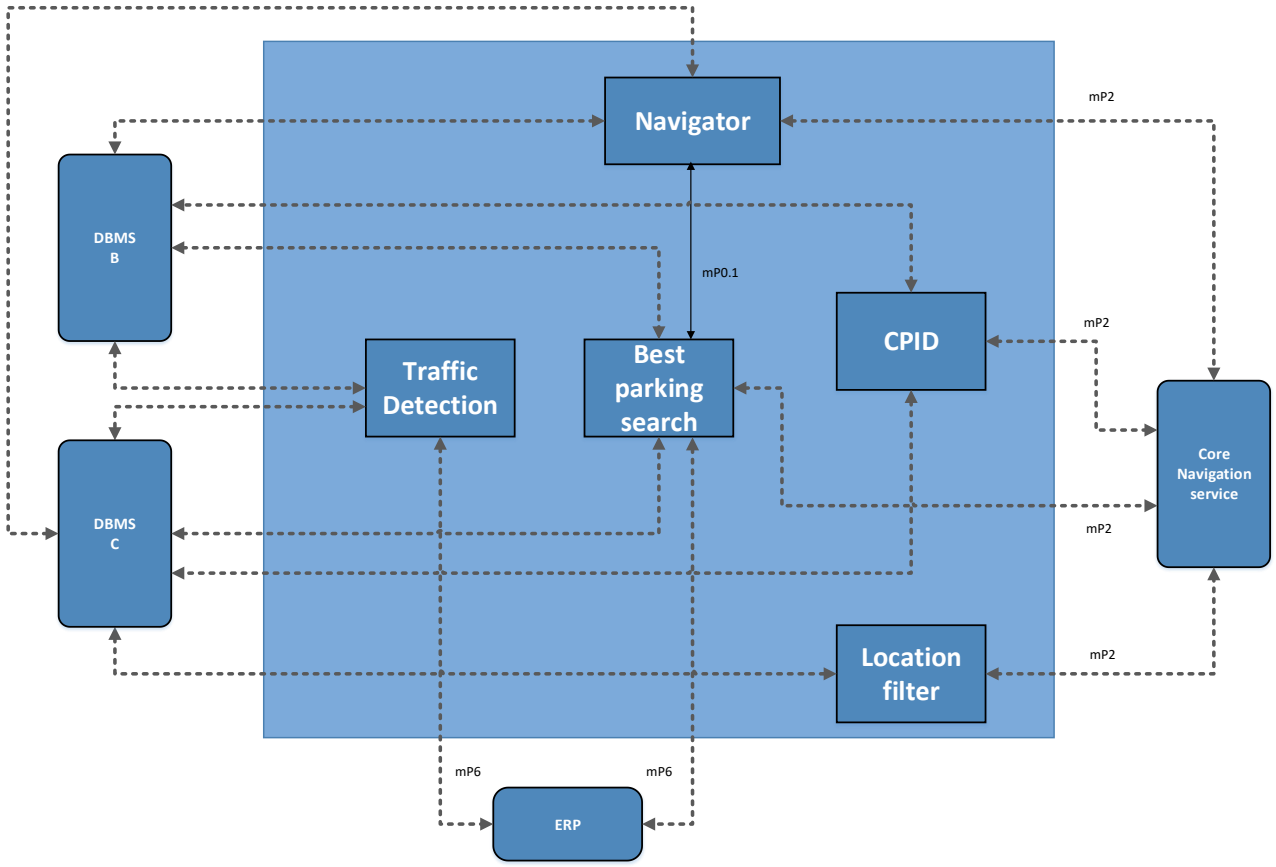


Message set e ruolo dei componenti invariato rispetto la soluzione 1.

Basi di dati e altre strutture dati:

- DB soluzione 1 invariati.
- Recent/frequent location DB, base di dati il cui ruolo è immagazzinare la posizione attuale dell'utente, le sue posizioni recenti

Architettura system level (soluz. 2)

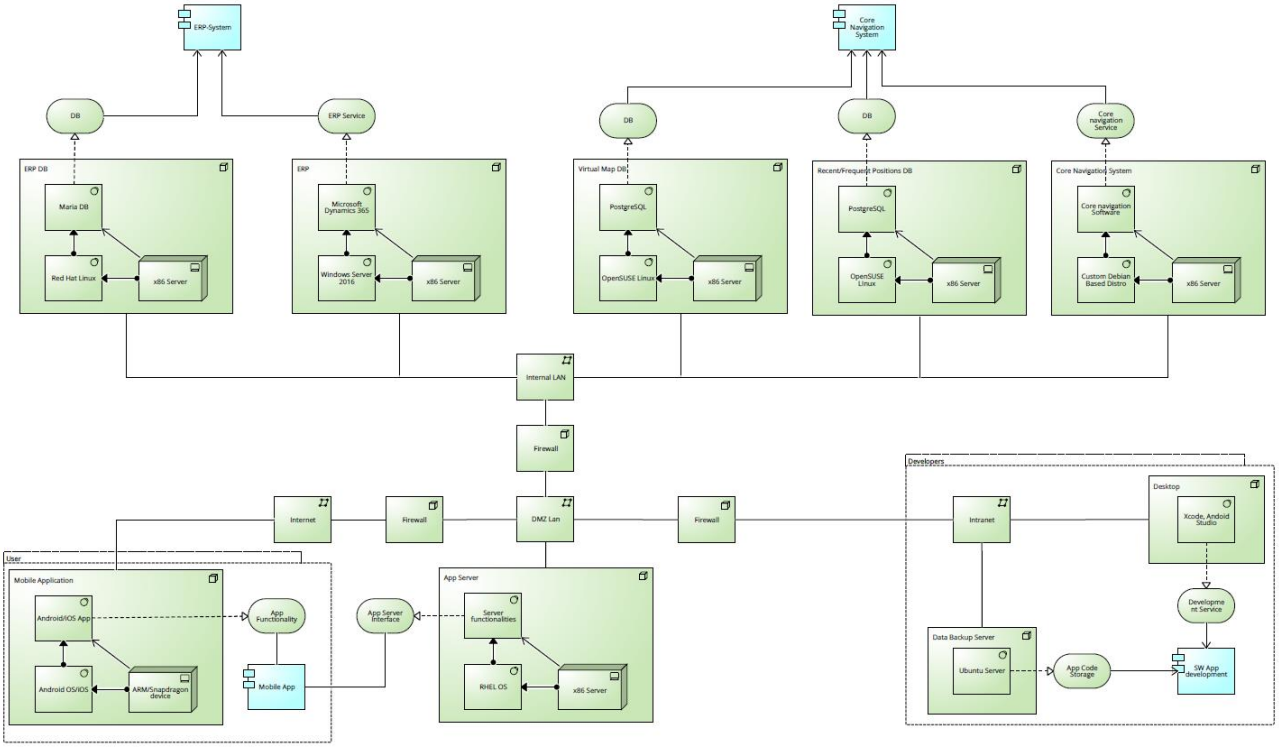


Message set	Content Exchanged
mP.01	Invio delle destinazioni relative al *miglior parcheggio* trovato da best parking search a navigator

Componenti del core navigation system:

- Navigator, le funzionalità di navigazione rimangono invariate rispetto alla soluzione 1, la posizione attuale dell'utente viene questa volta richiesta al DBMS C. Le destinazioni da raggiungere tuttavia vengono, ovviamente, ricevute dall'utente.
- Traffic detection, le funzionalità della rilevazione del traffico rimangono invariate rispetto alla soluzione 1, tuttavia le posizioni non vengono richieste al modulo navigator bensì al DBMS C.
- CPID (Customer Parking Intention Deduction), di nuovo le funzionalità rimangono invariate le posizioni vengono richieste non più al modulo software navigator ma al DBMS C.
- Best parking search, le funzionalità di ricerca del parcheggio e di verifica delle credenziali dell'utente tramite ERP rimangono invariate rispetto alla soluzione 1, la ricerca della posizione viene fatta attraverso il DBMS C mentre tra i messaggi scambiati con l'utente rimangono solo le richieste/notifiche di avvenuto parcheggio. Il modulo BPS fornisce indicazioni all'utente tramite navigator.
- Location Filter, nuovo modulo aggiunto nella soluzione, si preoccupa di interfacciarsi con l'utente e fornire le posizioni, che vengono slavate nel database C. Il modulo inoltre si preoccupa di catalogare le posizioni ricevute in recenti, attuali e raggrupparle come frequenti. Ciò consente il corretto funzionamento di tutti gli atri quattro moduli.

Architettura fisica (soluz. 2)



Le considerazioni per la descrizione del diagramma di ArchiMate relativo alla soluzione 2 sono analoghe a quelle già presentate per la soluzione 1.