

Solutions: End-to-end solutions

Advanced mobile services

THE MARKET FOR MOBILE SERVICES HAS DEVELOPED RAPIDLY OVER THE LAST FEW YEARS WITH THIS AREA QUICKLY MOVING FROM A "NICE TO HAVE" TO A CRITICAL "MUST HAVE" DRIVER OF REVENUES AND DEVICE MARKET SHARE. HOWEVER FOR MANY IN THE MOBILE INDUSTRY, EXTENDING BEYOND THE DEVICE REALM INTO A COMPLETE END-TO-END SERVICE OFFERING IS A COMPLEX CHALLENGE THAT REQUIRES A NEW APPROACH AND UNDERSTANDING OF WHAT CONSUMERS DESIRE, EASE OF USE, COMPLEXITY MANAGEMENT AND A RAPID LEARNING CURVE AROUND BACKEND SERVER/SERVICE TECHNOLOGY.



Market relevance of mobile services

The market for mobile services has developed rapidly over the last few years with this area quickly moving from a "nice to have" to a critical "must have" driver of revenues and device market share. However for many in the mobile industry, extending beyond the device realm into a complete end-to-end service offering is a complex challenge that requires a new approach and understanding of what consumers desire, ease of use, complexity management and a rapid learning curve around backend server/service technology.

We can trace the start of the service revolution to the mid-nineties when we first started to send picture messages using Nokia's picture messaging platform. This was arguably the first time consumers started to use more advanced custom services enabled by GSM. This slowly evolved through many stop-start efforts, typically based around SMS and basic gaming services before Nokia upped the ante with the launch of the N-Gage service. N-Gage was for many the first true comprehensive end-to-end service that tried to enable a complete third party ecosystem. While N-Gage met with limited success it is fair to say that it put the services agenda on the map. Then came iTunes and the iPhone, and the new mobile service revolution was truly born.

As we look forward we see a new revolution for mobile services on the horizon. However this is not one of consumers embracing the use of services but rather where they will use them. Over the past few years the focus has been firmly on the traditional mobile phone market, but now we see many more devices coming into scope. For example, Netbooks, eReaders, In-Car entertainment systems, In-home entertainment systems. Portable games consoles, etc all look set to leverage mobile services as a core part of their offering. And as consumers become more sophisticated in their use of such services they will expect to be able to switch from one device to another and enjoy a seamless service provision experience.

In this white paper we attempt to explore many of the key challenges and aspects in designing, implementing and managing a mobile service experience including providing reference architecture examples and lessons learned from Teleca's many years of experience working in this domain.

Andrew Till
November 2009

INTERNET BASED MOBILE SERVICES

Figure 1 gives an overview of the main use cases that are currently seen as the key drivers for mobile services. They range from simple search functions to advanced services to access legacy systems that make enterprise tools and data available on mobile devices.



Figure 1: General mobile service use cases

In the past, mobile services were mainly seen as the access to the World Wide Web, and basic services such as email from the mobile device using either the web browser or dedicated email applications. The user experience was often disappointing, lacking an appropriate user interface design and not yet optimized for constraints of smaller screens and slower, and often interrupted, data sessions.

With evolving technologies, improvements in the quality and speed of UIs and availability of affordable data plans, the situation changed dramatically in the last few years. New development environments such as the Web Runtime from Nokia facilitate and speed up development of web applications and provide access to dedicated functions (such as GPS and the camera) from the underlying platform. Application stores offer an attractive distribution channel for developers as well as a trusted source for consumers to discover and purchase applications.

Services have also moved from being an adjunct part of the consumer offering to form an integral part of the overall value proposition. This is true for device manufacturers as well as service and content providers as they all aim for a compelling integration of their services and offerings into the overall device experience. This includes the seamless integration of social contacts to the device address book and the possibility to communicate with all contacts in a unified way independent of the type of contact. With the push for mobile services coming not only from device vendors and network operators but also from many traditional web service companies as they seek to extend their reach and monetization potential, we are seeing consumers embracing these new experiences across the market segments and age ranges.

SERVICE PLATFORM OVERVIEW

In this section we provide an overview of the different options for developing an end-to-end service, looking at both the client side (on device) and back end (off device) technologies and design considerations. Figure 2 gives an overview of the building blocks of modern internet based mobile services. Besides the actual service implementation within the provider's service domain, they rely also on device capabilities as well as on further supplementary services delivered via other service providers. Perhaps one of the key trends over the past twelve months has been the recognition that not all services presented to the consumer need to be owned, developed or defined by the core service provider. Focusing on core services and then providing standard APIs for third party services that follow the core user experience paradigm is becoming a popular approach to the market. While this approach may reduce the overall level of control the solution provider has over the overall offering, it does allow for a number of significant advantages such as:

- Reduced time to market for supporting new web services
- The ability to support niche consumer segments early
- Reduced overall costs versus a purely internal development strategy
- Greater leverage of external innovation

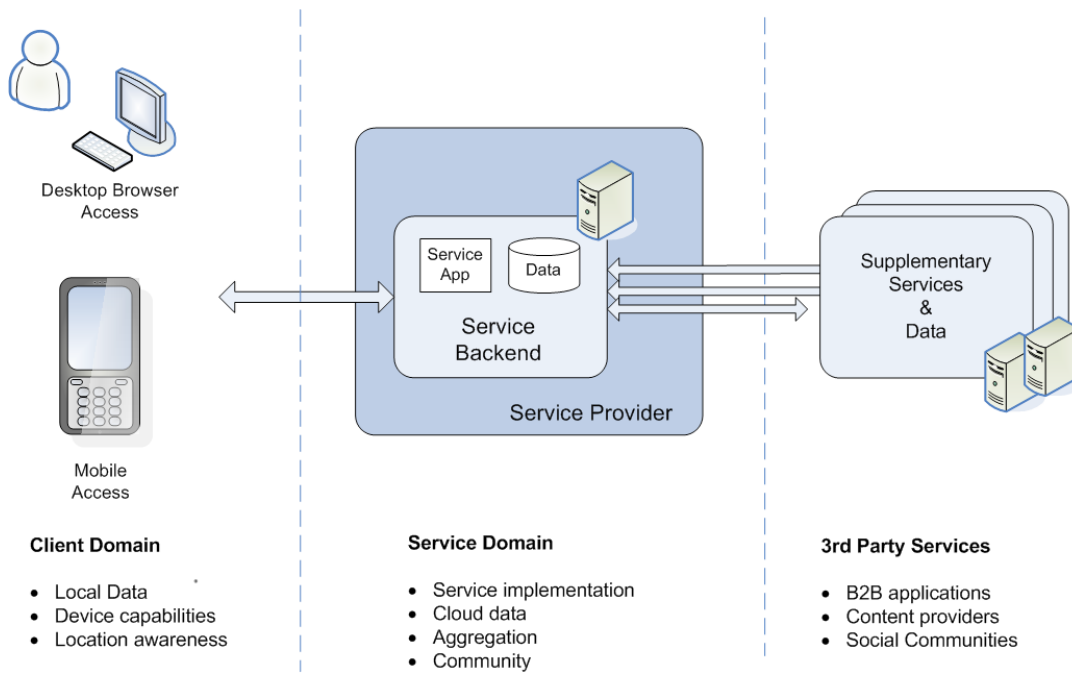


Figure 2: Service domains

Client Domain

Leveraging the specific capabilities of mobile devices will be one of the differentiating factors of mobile services compared to 'plain old' web applications. New services are starting to utilize device capabilities such as GPS to offer location-based services like navigation or location based service discovery. Further device capabilities include:

- User data (address book, calendar)
- User generated content
- Camera
- Media player and native codecs
- Device sensors
- Presence information

We expect that the combination of device capabilities and existing online services will create many new and exciting services as well as being used to extend existing ones.

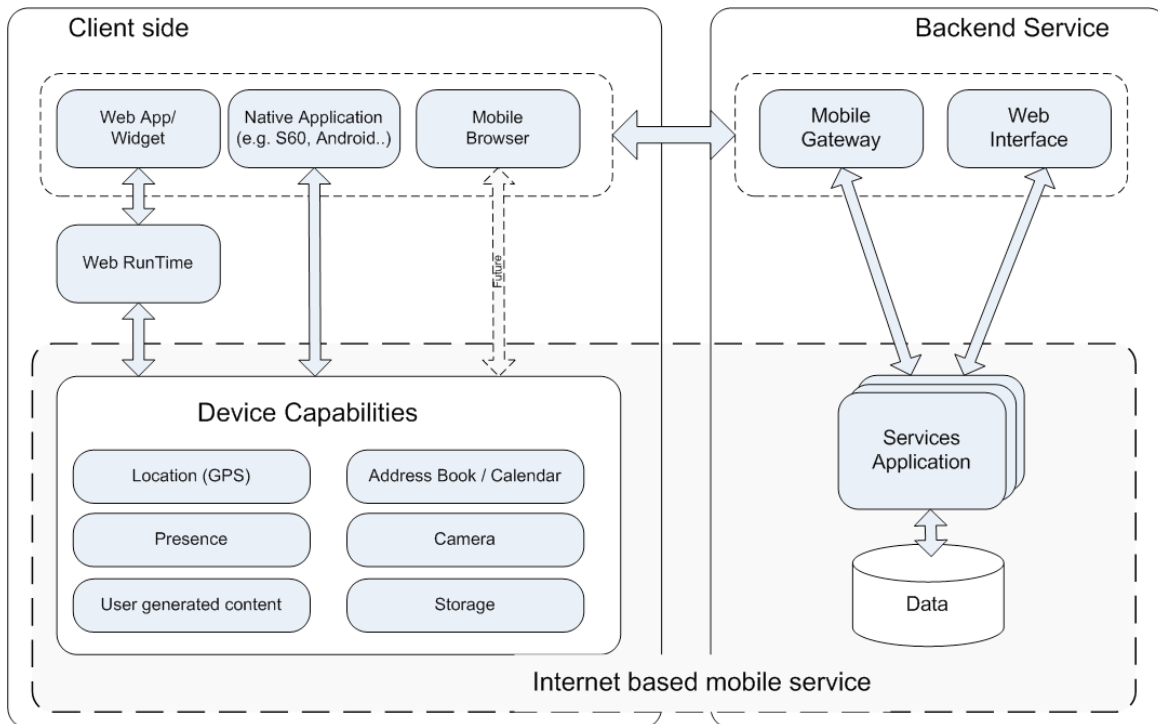


Figure 3: Service composition from client and backend features

As outlined in Figure 3, there are three fundamental design alternatives for the client-user interface and application implementation:

- Web application (via mobile browser)*

The user interface and application code is implemented on the server side. The device browser ‘only’ renders the delivered html pages and potentially executes some JavaScript code delivered by the server. In this model the application has no special access to enhanced device capabilities.
- Widget application*

User interface and application code (based on standard web technologies) is running as a kind of stand-alone application on top of a web runtime environment. Data is fetched from the server and then processed and displayed by the widget. In this model the application can take advantage of enhanced device capabilities that have been exposed to the web browser and runtime environment.
- Native application*

User interface and application code is realized in a native language such as Android Java, Symbian S60 C++ or J2ME. Similar to the widget approach, the data is fetched from the server and processed and displayed by the native application. In this model the application can typically take advantage of a wide range of enhanced capabilities that have been exposed to runtime environment (for example in J2ME via the provision of specific JSRs).

The table below provides an overview of the pros and cons of the different approaches that can be used for client applications.

Design Alternative	Pros	Cons
Browser based application	<ul style="list-style-type: none"> • Uses standard web technologies (HTML, CSS, JavaScript, XML), which facilitate application development • Deployable on ‘any’ mobile device supporting a standard web browser • Changes to UI and application features can be deployed at any time from the server 	<ul style="list-style-type: none"> • Currently no access of standard browsers to device capabilities such as GPS, calendar, file store etc. • Very limited local storage • Subtle differences in the browser implementations on various platforms which can result in inconsistent user experiences on different devices
Widget application	<ul style="list-style-type: none"> • Uses standard web technologies which facilitate application development • Enables (limited) access to platform capabilities via JavaScript APIs provided by the runtime environment • Can be fast to develop and deploy to consumers 	<ul style="list-style-type: none"> • Available platform functionality depends on the runtime environment • Widget environments are currently not standardized leading to a fragmented market • Widget environments available only for selected device families and OEMs - typically mid to high tier devices
Native OS application	<ul style="list-style-type: none"> • Full access to platform • Seamless integration of services to the platform is possible • Applications can be optimized to specific platforms for improved performance • Easier to create customized APIs if core functionality is not supported by the platform 	<ul style="list-style-type: none"> • Needs creation or porting of the application to every target platform OS (and potentially OS versions) • Requires specific platform knowledge and development time can be longer • Applications typically have to pass specific platform testing criteria (for example as Symbian Signed or Java Verified)
Java based (J2ME, Midlet)	<ul style="list-style-type: none"> • Enables applications running on a multitude of platforms and device families • Many standard features supported by the J2ME engine, such as garbage collection, to reduce application size 	<ul style="list-style-type: none"> • Subtle differences in supported functions still require testing and porting effort • Many JSRs are optional resulting in inconsistent J2ME engine implementations

Table 1: Pros and Cons of client side design alternatives

Backend Domain

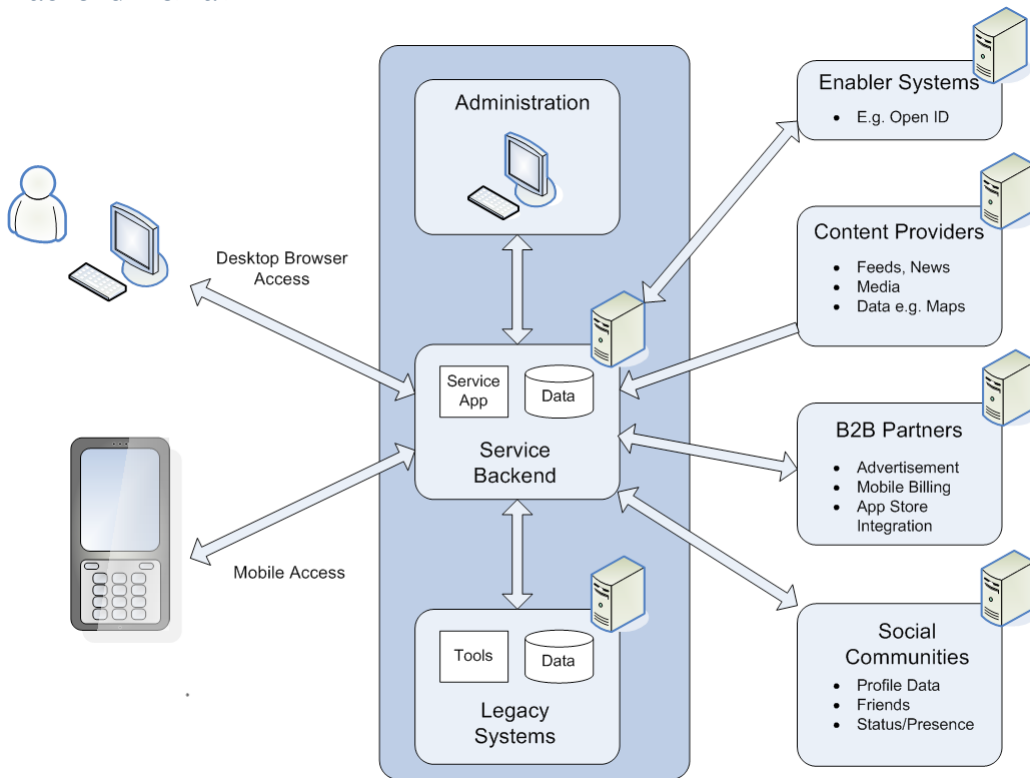


Figure 4: Mobile service landscape

On the backend side the service usually consists of a core component which realizes the actual service implementation and provides the service entry points (interfaces) for the various client representations. Depending on the type and complexity of service several further components might be in the scope, namely:

- *Company legacy systems*

Legacy systems are existing network operator or enterprise tools and databases which can or should be leveraged on the service provider side. Examples for those reusable components are stock and customer databases or existing internal services for customer relation management or resource planning.

- *Service enablers*

Service enablers support the general service infrastructure with dedicated services such as Open ID authentication, which enables user authentication across different services using the same digital identity. Another example is geo-location services which allow the estimation of the user's location based on their IP address to provide tailored services or UI representations. By using service enablers you can remove duplication and enhance the overall consistency of the service presented to the user.

- *Content providers*

Many web services today rely heavily on other existing services or data in the web. They reuse or combine existing services forming a new offering (a mash-up) with a unique and added value for the end customers. Examples are Google Maps service as well as media databases and feeds for traffic and transport or news services (referred to as layers) which are rendered on top of the core map image.

- *Business-to-business partner systems*

There are two main business cases for mobile and online web services. Either you have assets or services to sell or you build up a big community of users around a (free) service and try to monetize the user base via advertisements. In both cases you would usually rely on business partners to integrate ad services or mobile billing to your infrastructure.

- *Social communities backends*

Depending on the type of service offered, the interaction with existing social communities might be a crucial part of the new services offering. Such interactions may include access to existing profile data, friend lists and status information provided by the community service via social APIs. In turn they usually allow the update of information allowing the synchronization of status and presence information among different services.

This allows for aggregation services which enable the creation of a single unified messenger or community application which uses only one backend infrastructure that aggregates and synchronizes relevant data across different networks.

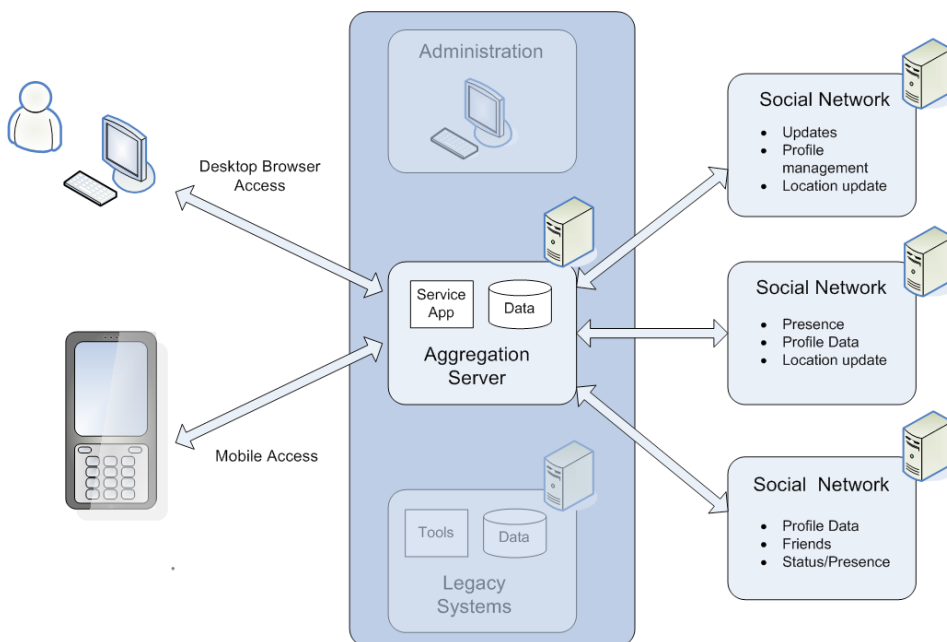


Figure 5: Aggregation of services

Figure 5 shows the idea of aggregating content from different sources. Social communities such as Twitter, Facebook and MySpace enable access to profile data and status information through RESTful (or REST-like) APIs. As they are not standardized, client applications rely on service specific client libraries to interact with the individual services. The aggregation of multiple services and data sources on the server side avoids these dependencies on the client side and provides a single integration point for multiple services. There are a number of benefits of such an approach as follows:

- Services are presented to the user in a consistent manner
- All new services supported are easy to find and use for the consumer
- Minimize data usage, as associated costs, as the client only communicates with the aggregation server and reduces overheads
- Ideal for pull based/timed based offerings – updates from the different services can be combined and stored on the aggregation server and sent to the user as a single payload when they pull the data down.

PC Client

Although we are focusing on mobile services it is still applicable to look at the PC as a key part of this experience as most consumers also use a PC on a regular basis. It should not be forgotten that most consumers will use the largest display available to them for a given application. It is also worth noting that while flat rate data services have become more common they still remain far from universal and in many countries it is more cost effective for the user to perform certain tasks on the PC. For activities such as content purchasing, many publishers like to provide enhanced content overviews for PC users. Therefore we believe that the PC should still be viewed as an integral part of the overall mobile service experience. Typically a PC client is used to support the following:

- Off-line synchronization of data
- Large media file transfers
- Media management
- Back-up of user data
- Content discovery

GAMING PLATFORM EXAMPLE

In this chapter we attempt to show what a commercial service offering may look like. Given the current popularity of mobile games we have chosen to base this example on a mobile gaming service offering. For a more detailed discussion on this subject please refer to our White Paper on Advanced Mobile Gaming Services. Figure 6 shows the core building blocks of a mobile gaming platform as an example of a complex end-to-end service architecture. The starting point here is the main end user cases defining the mobile gaming experience:

- Discover, purchase and download games (via PC as well as over the air)
- Activate games or specific game features
- Manage games, profiles and friends on the device
- Share your experience with friends and online-community

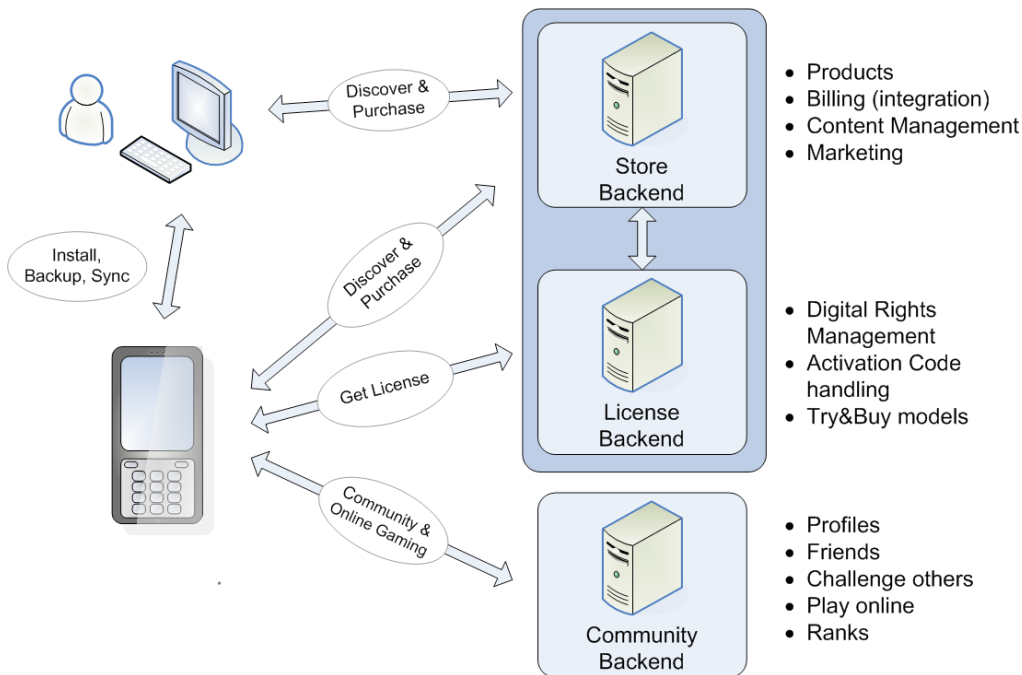


Figure 6: Main building blocks of a mobile game platform

Client Architecture

Overview

In Figure 7 below we assume that the games platform provider decided to offer a games management framework, which encapsulates the games into a common frontend tool (here Games Manager Application). This enables a consistent overall look and feel for the games platform from the user perspective and presents all games related functions through a single entry point.

We assume that an abstraction layer (Gaming Libraries - for a more specific discussion on this subject please refer to our Advanced Mobile Gaming Services white paper) would hide OS specifics of the mobile platform capabilities from game developers. This allows:

- Application development in standard C/C++
- Easy porting of games from/to other platforms
- Binary compatibility through a complete range of device models

While maintaining full access to platform capabilities such as graphic acceleration, presence information, geo-data, user data, etc.

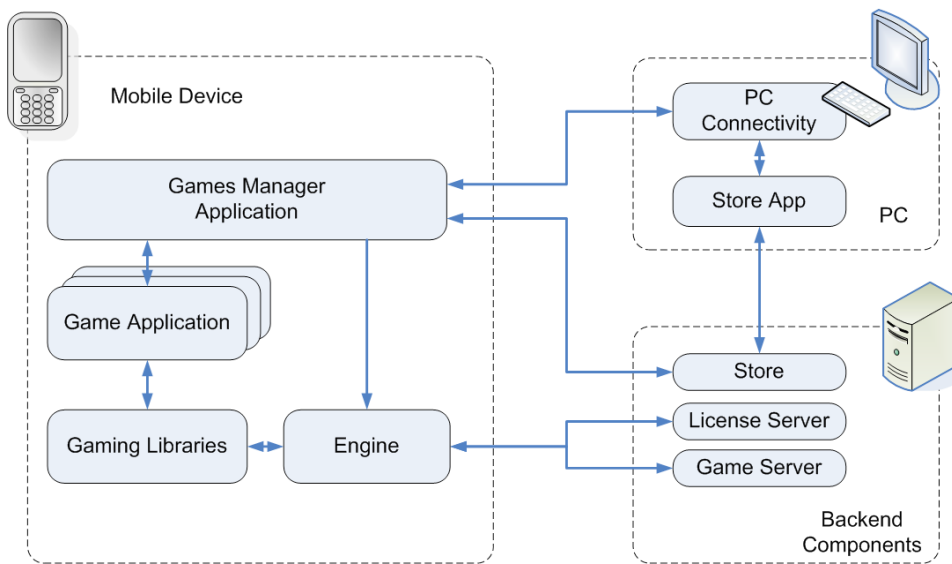


Figure 7: Mobile device core components

Many of the features offered through the platform depend on services or information available on the backend. The Engine component in Figure 7 represents all supportive functions which enable service integrations with the on-line store, games and license backend.

As an alternative to the above approach, game applications could also be realized as stand-alone (native OS) applications, which could then communicate directly with the backend components. While the effort for the platform provider is significantly lower, most of the needed integration work is simply shifted to the applications and repeated for all games again and again. Native applications are tightly coupled to the underlying OS, which makes porting more difficult. Depending on the scope of the platform and the business criteria a hybrid approach might be applicable.

Games manager and store application

The game manager application manages all aspects of game application handling and offers a seamless integration to online services from the end-user perspective. It also presents a consistent look and feel and ensures that the gaming platform is perceived as a uniform offering. The following features are typically supported:

- Game installation and management on the device
- User profile management
 - Personal information, ranking, presence status, mood
 - User icons, avatars
- Access to community features
 - Friends list
 - Instant messaging and game invitations
 - Rankings
 - Online gaming
 - Game ratings and recommendations
- Interworking between games and games manager
 - In-game purchase (activations of trial applications and game extras)
 - Game meta-information handling (score, extra items, etc)

Online store related features:

- Game catalog browsing
- Access to additional marketing material (screen shots, videos, teasers)
- Purchase and download games
 - Variety of billing methods (credit card, operator billing, online billing providers)
 - Variety of license options
 - One-click buy (saved billing information)
 - Confirmation email

Interaction with the backend

The store application interacts with the store backend during discovery and purchase transactions. Depending on the desired integration level, the device side store representation can be realized as:

- Stand-alone or embedded browser application,
- Widget application based on a web runtime or

- Native implementation within the Games Manager application

Embedded browsers, if supported by the mobile platform, offer a good compromise between flexibility and integration levels. The game catalog product pages can be created dynamically as (X)HTML or Flash® content while preserving the overall look and feel of the application.

Game Service/Store Backend Architecture

The main purpose of the store backend is to present games to consumers for downloading and purchase. Often the complete web content is managed by the same backend infrastructure and provides a consistent look and feel, even for marketing content

Core functions

Let's have a more detailed look at the system components required on the backend side. Besides the core components mentioned above (online store, license and community backend), several further components from the existing enterprise IT infrastructure usually need to be integrated. Examples could include existing content management systems (CMS), separate reporting or resource planning modules as well and customer care tools.

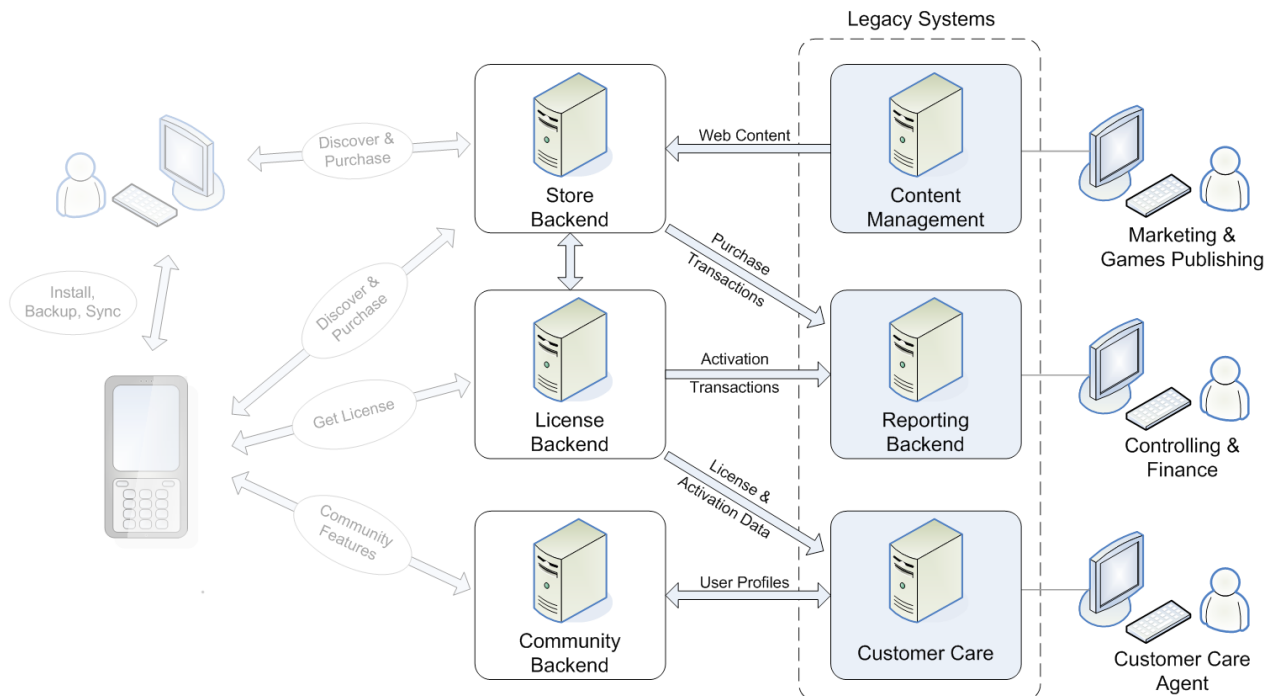


Figure 8: Core backend functions

Those sub-components may be separate software components deployed on dedicated server hardware (possibly distributed over several company sites and organizations) or can be part of an integrated solution. In a multi-million user environment, most front-end components tend to be modular components, which can be distributed to several servers. Legacy and more enterprise level systems are often separate entities as they are usually managed by different teams or organizations.

Depending on the original business requirements and existing legacy infrastructure, either customized self-made solutions or mature off-the-shelf third party solutions may be selected for any of the listed modules.

On-line store

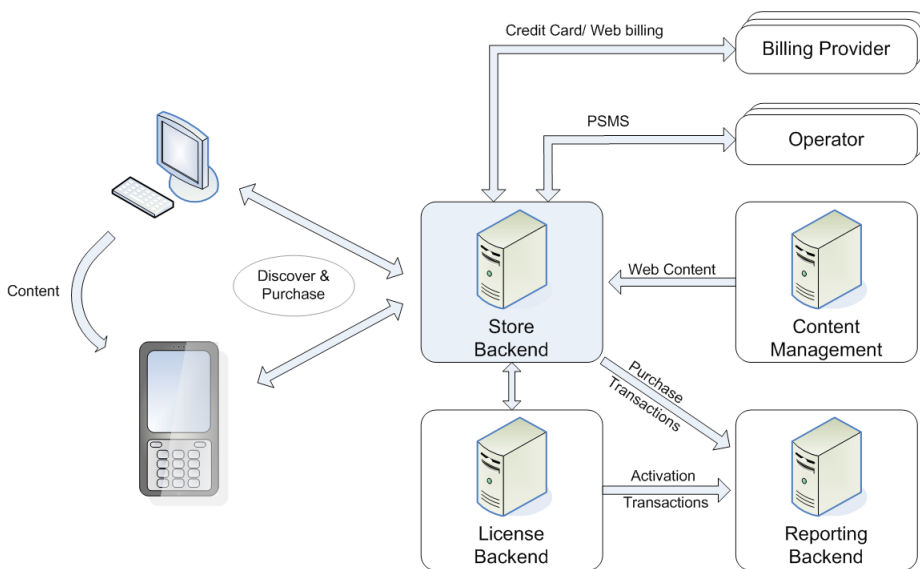


Figure 9: Store sub-system

The on-line store presents the game product catalog to consumers for downloading and purchasing. The web marketing assets (such as product pictures, screenshots and video-trailers), content files and pricing and categorization information needed for the online shop are provided by the Content Management System. During the purchase, the backend interacts with the license backend to create the related license object and potentially an associated activation code.

In Figure 9 we assumed that third party billing providers would be integrated covering credit card payment, online payment (such as PayPal) and operator billing. After the consumer has confirmed the purchase, the activation code or license object respectively will be transmitted and installed to the

mobile device. Finally, a transaction log is sent to the reporting module after the completion of the purchase.

License backend

The license backend is the central component for issuing and managing game licenses and activation codes. Within the scope of this document we assume that a license grants access to protected content (e.g. game level files). Depending on the digital rights management scheme used, such a license usually includes encryption keys and the usage rights description for the encrypted content.

The license backend component has the following key roles:

- Creation and issuing of valid license keys to consumers (based either on a sales transaction or provided proof of purchase)
- Creating and issuing of activation codes to business-to-business partners and on-line stores
- Storage of license (purchase) history to enable reactivation
- Re-issuing of license keys in the event of device damage or content loss
- Online validation as fraud prevention mechanism

Depending on the actual DRM technology and schema used, the requirements for mobile devices and the license backend components vary significantly. In mobile environments commonly used frameworks are listed in the following table.

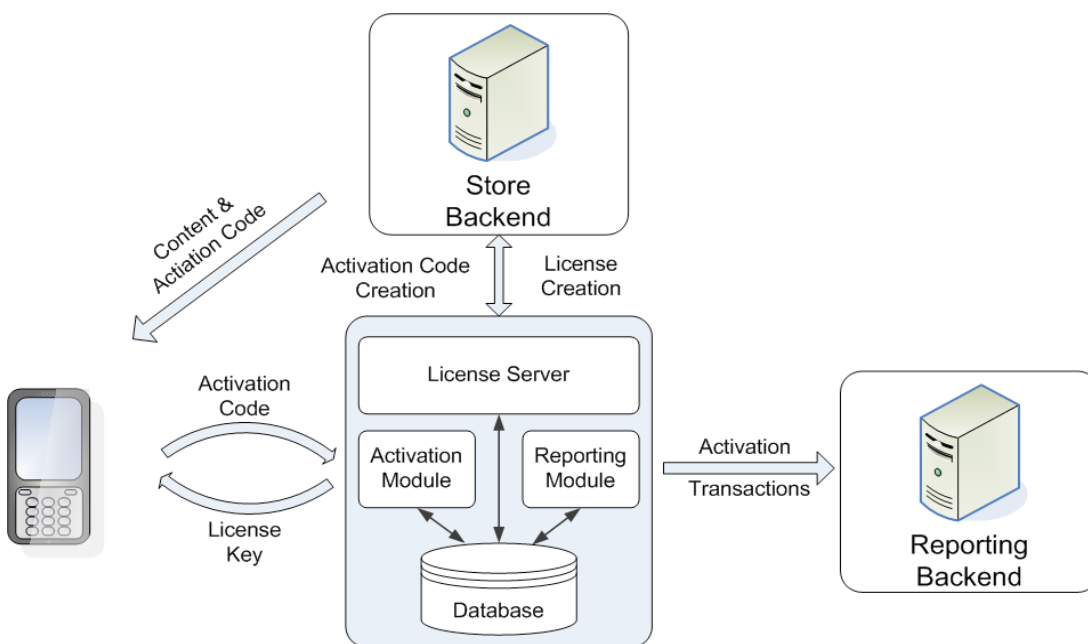


Figure 10: License back-end sub-system

Online community

The social aspect of gaming services is gaining more and more importance and popularity with users. It makes the platform much more attractive as it enables users to find new challenges as they improve their skills. Additionally, with growing reach, the brand awareness of such services and enhanced user loyalty to the platform can help in monetizing the service. Community features typically cover:

- Friends lists
- Game invitations
- Rankings and tournaments
- Game ratings and recommendations
- User discussion forums (latest games, devices, features)
- Customer care (FAQ, technical support)
- Blogs

Many of the features should be available in the web offering as well as from the mobile device directly. Ideally, the services would be integrated into a common tool like the game manager application above, or directly available to the games applications based on dedicated local APIs or on-line web-services.

PC and Web Experience

Discovery of games in the application store can be done from either the mobile handset directly or using a standard browser application on the PC. Although the capabilities of the target platforms are quite different, the look and feel of the store and the purchase experience should be consistent through the different channels, enabling users to quickly find the content they want irrespective of the channel they are using.

Usually a PC connectivity tool supports the user in the installation process and enables data backup and restore functionality from the PC.

SUMMARY

This paper set out to provide the reader with an overview of what is involved in developing and delivering a compelling end-to-end service experience. To provide an understanding of the alternative options available for development and to give a practical example of what a real life service could look like. In this summary we seek to share some of the key learnings we at Teleca have derived from developing both commercial services and advanced prototypes for our customers.

System Design Challenges

Requirements management

In theory, designing and building end-to-end services should follow a very structured process ranging from initial requirements gathering to the development of complex specifications and interdependency models. And this would have been the case just a few years ago. However, modern web services have evolved and become successful by thriving on speed and responsiveness to consumers and gained first move advantage. This is highly reflective of what is now happening in the mobile services domain with solutions often being deployed in an environment which spans multiple external companies, often start-ups, internal units, different countries and time zones, and it's often difficult to maintain a working requirement management process based on traditional approaches.

Finding a requirements and change management process is easy to understand (and remember) and that it defines clear responsibilities is in our view a key success criterion. Recognizing how, in reality, changes will be made through the development cycle will be integral to a successful project. Exploring Agile or Scrum based development methodologies may also be more appropriate especially for Services when you plan to test incremental service enhancements on a regular basis in order to rapidly identify what resonates with your target consumers.

Design for Test

It is essential that system integration and system validation testing is a significant part of the effort needed to make the whole end-to-end environment described in this paper work. In early design phases the system testing is often neglected if not ignored completely.

This repeatedly leads to situations where specific functions, error situations or environments cannot be tested at all or only with an unreasonably high effort. One example is error situations between interacting sub-systems during system integration testing. If the service backend components work as specified (which is certainly desirable), specific errors cannot be tested. Those errors however might trigger specific routines on the client side, which also need to be validated in an end-to-end fashion.

So system validation and integration testing should always also be considered in the early design phases where it is much easier to include essential test support to the components under design.

Further key points for system testing are:

- Testing execution needs to be aligned to component releases
- In an end-to-end environment, client and server component software release management need to be aligned
- Acceptance of a system component can usually only be performed against a reference. This reference is a certain requirements baseline agreed on a specific date of time.

Core Skills Required

Overall system design:

- End-to-end system architecture design
- Information architecture design
- Requirements and change management

Client related skill sets:

- User interface and user experience design
- Software design on target platform (native code or Java ME)
- OS independent API design
- Web based technologies (Widgets, Webkit, Ajax etc,)

Backend related:

- Server side technologies (J2EE, .NET, Server side scripting etc)
- Multi-tier backend system design
- Interface design
- Scalability and performance
- Testability

Final Thoughts

The market for mobile services has been radically transformed over the past two years. It is no longer enough for mobile device offerings to simply focus on the handset but they must now include the complete end-to-end user experience. This trend is set to become stronger and to permeate into many other market segments outside the traditional mobile phone sector. Perhaps the most challenging and daunting issue for many of us is how to adjust to the pace of this market and the evolving myriad of development options available. At the same time it provides a bridge to a wider range of consumer touch points, and as new segments are penetrated, a wider range of monetization points.