MODEAS: TOWARDS A MODULAR ENTERPRISE APP STORE

LISARDO PRIETO-GONZÁLEZ
SRH Hochschule Berlin, Germany
lisardo.gonzalez@srh-hochschule-berlin.de

RICARDO COLOMO-PALACIOS
Østfold University College, Norway
ricardo.colomo@hiof.no

BEATRIZ PUERTA
Carlos III University, Madrid
beatriz.puerta@uc3m.es

GERRIT TAMM
SRH Hochschule Berlin, Germany
gerrit.tamm@srh-hochschule-berlin.de

Mobile technologies and more precisely apps have disrupted technology scenario. Bring your own application (BYOA) is a new trend that has emerged after the "bring your own device" (BYOD) vogue. However, and in spite of its intrinsic benefits, the trend presents also several caveats and Enterprise app stores are seen as a way to tackle these risks. In this paper, authors present ModEAS, a system designed to serve as a modular and scalable architecture when developing a middle scale or personal Enterprise app store.

Key words: Application store, mobility, personal devices, mobile applications

1 Introduction

IT has been considered fundamental for the development of productivity and knowledge-intensive products and services (Soto-Acosta, Martinez-Conesa, & Colomo-Palacios, 2010). Radical changes have marked the mobile phone industry since the first commercial 3G mobile phone was launched by NTT DoCoMo in 2001 and the emergence and rapid growth of smartphones have instituted radical structural changes (Hsieh & Hsieh, 2013). Thus, mobile service business has moved into a new epoch due to the explosive growth in mobile application (‘‘app’’) services available at ‘‘App Stores’’ (Kim, Park, Kim, & Lee, 2014). In spite of its apparent novelty, these are not the first attempt to provide mobile content; for instance n I-Mode launched in 1999 can be considered a valid precedent (Cuadrado & Dueñas, 2012). Nowadays the two leading approaches in the area are Apple AppStore for iOS and Google Play (formerly known as Android Market). The Apple App Store is a digital application distribution platform or open market for iOS developed and maintained by Apple while the later was a similar proposition developed to distribute Android apps. However, the landscape seems to evolve. And one of these innovations is the trend towards Enterprise or Corporate App Stores. According to Gartner (Finley, Redman, P., Prentice, B., & Buchanan, S., 2013), by 2017, 25 Percent of Enterprises Will Have an Enterprise App Store and for 2013 Enterprise App stores is one of the trends to watch for CIOs (Costello & Prohaska, 2013) as well as the whole IT industry (Gartner, 2012). Also, software vendors are getting more and more interested in cloud oriented mobile applications (Colomo-Palacios, Fernandes, Sabbagh, & de Amescua Seco, 2012). In any case, in the next future Gartner believes that many organizations will deliver mobile applications to workers through private application stores (Finley et al., 2013; Gartner, 2012). Enterprise app stores promise at least a partial solution but only if IT security, application, procurement and sourcing professionals can work together to successfully apply the app store concept to their enterprises (Finley et al., 2013). Bring your own application (BYOA) is a new trend that has emerged after the bring your own device (BYOD) phenomenon that can be viewed as a symptom of the so-called 'consumerisation' of IT' (Gedda, 2012). If employees perceive that they can do their jobs more effectively using an application downloaded to their own tablet or smartphone, then they will simply expense it, giving rise to the BYOA trend (Walters, 2013).

In spite of its multiple concerns (Miller, Voas, & Hurlburt, 2012; Morrow, 2012; Thomson, 2012), BYOD and BYOA are a fact that almost none can ignore. In this new scenario, Enterprise App stores can increase the value delivered by the application portfolio and reduce the associated risks, license fees and administration expenses (Finley et al., 2013). However, Enterprise App Stores also present some challenges, for instance, according to Gartner, (Finley et al., 2013), a dynamic selection of apps is mandatory to keep users interested in visiting the store.

Given that scenario, authors present ModEAS, a modular and scalable architecture to consider when developing a middle scale or personal Enterprise app store, as well as some use cases and issues to have into account depending on its usage context (or range), such as rich Internet applications (Colombo-Mendoza, Alor-Hernández, Rodríguez-González, & Colomo-Palacios, 2013) for clients and administrators. The novelty of the proposal is based on the scalability and adaptability of the application, ranging from enterprise-wide to personal app stores but also on the security of the data that can be recovered by means of software mechanisms.

2 Architecture and implementation

Instead of creating a closed system, authors decided to analyze and identify the main different elements involved in an application store and develop a set of functionalities, following best practices such as object-oriented design, to create flexible and extensible modules with low coupling and high cohesion. This resulting set of functionalities is provided as an API (Application Program Interface) that can help to create a wide set of tools, such as a personal application classifier/organizer, a web based application store or a desktop application for remote management of that web application among others.

An application store is intended to distribute software in a free or paid manner among a narrow or wide set of users, so in a first approach, it is possible to identify two main elements in the system: **applications** and **users**. Requirements are as follows:

- **Security**: the system should be safe from hackers, preventing data leaks or license thefts. Considering that ModEAS could implement a pay system, and it could store sensitive data about the registered users, it is a very important issue.
- **Reliability**: this point has two meanings. Related to technical aspects, the system should provide a mechanism to prevent data loss or corruption. In a mid-scale Enterprise or in a personal environment, a RAID (Patterson, Gibson, & Katz, 1988) may not be available, so a software method to mitigate corruption or data loss based on existing solutions was designed. Related to the user of the application store, is very important to establish a trust in the system. In this case, ModEAS offers the possibility of establish secure communication methods between the store and their devices (i.e. creating SSL connections between them).
- **Scalability**: the described architecture is oriented to mid-scale application stores, but depending on several factors, such as a high growth of the company in which it is applied, the number of final users could increase dramatically. The system should be able to support this situation.
- **Flexibility**: the designed system is aimed primarily to the distribution of mobile applications, but it should be easily modified to support computer software licenses, books or multimedia content among others.
- **Efficiency**: although the system is designed for a small number of users (order of hundreds), it should run in a cheap environment, making optimal use of resources wherever possible. This will help in the previously presented scalability scenario.
- **Portability**: the system should run on a different number of heterogeneous Operating Systems.

To achieve portability objective, Java™ programming language was chosen in order to create the application core, which is a set of objects and methods that provide enough functionality as to create a personal application store and/or similar related utilities. It was decided to use Java™ due it is a widespread and proved development platform that offers several benefits, such as flexibility, efficiency, cross-platform compatibility, built-in protection from viruses and cracking, among others.

When designing the application core API, several functionalities were considered concerning both the users and the applications. *Users* are abstract entities representing people (or people devices) registered in the system. These *users* can have two main profiles (*normal* and *administrator*); although some other profiles could be defined and configured (i.e. *content_developer*). Users are basically defined by a *unique ID* and some *login credentials*, depending on the required security. Also this definition can be extended by applying a custom set of attributes, such as *device_id*, *first name*, *second name*, *address*, etc. All this attributes provide additional information if it would be necessary to implement user profiles.

In several cases, it could be interesting that a set of *users* could have access to certain applications, but not to other ones. Also it could be possible to assign a set of licenses to some *users* (for example, working in the same department). To cover these needs in a structured way, the element *group* was defined. Basically a *group* is an abstract entity composed by a set of *users* and a set of common properties among these *users* (allowed and denied applications, assigned licenses, downloaded applications and devices in which they downloaded these applications). A *user* can belong to several *groups*. In that case, it will have the less restrictive set of properties from the properties inferred by all the groups it belongs (this is, access to applications, assigned licenses, etc.).

Applications are abstract entities representing the content that *users* want to get either by paying or free. In the initial design, the architecture was based in a mobile application store (that is why term *application* was used), but this can be easily extended into general digital contents (in that case, the term *content* instead of *application* should be used). An *application* is defined by a *unique ID*, and by its *physical location*. It also has several attributes, such as the *application name*, *version*, its *size*, *permissions required* (i.e. send short text messages, access the physical storage, etc.), *price*, *license type*, among others. These attributes can be defined and extended, because they depend on the represented content (obviously, it is not the same an iOS application that an Android application).

Concerning to security, reliability and efficiency issues, there have been defined a couple of functionalities that affect the definition of application. These new functionalities involve the addition of two new (optional) attributes in the definition of *application*. Firstly and regarding data integrity, ModEAS implements several functions to calculate hash sums of the stored contents. These hash sums can be stored as an additional attribute. Giving the nature of the contents, ModEAS implements several methods to calculate the checksums: MD5 (Rivest, 1992, p. 5), SHA-1 (“FIPS 180-1. Secure hash standard,” 1996), SHA-256 and SHA-512. The usage of this functionality is optional, as well the *hash* attribute is. Secondly, regarding data recovery, in systems not implementing RAID or any other data redundancy methods, it was decided to provide data recovery based on PAR2 (Nahas, Clements, Nettle, & Gallagher, 2003). This can have an extra computational load as well as a slight increment in the size of the stored *application* definitions (due the new attribute *par2* which includes PAR2 sums for each application indexed in the repository). Apart from par2, ModEAS API can provide also support to high-end turbo codes (Ismail, Douillard, & Kerouédan, 2013), although these codes are not implemented yet. As in the hashing case, the usage of this functionality / attribute is pure optional, and it is just intended to provide extra data reliability to the implemented system.

Back to users, about roles, depending on the profile, a *user* can do more or less actions. By default (but depending on the implemented features), the *administrator* is able to:

- Add applications into the system
- Manage system applications (classify, rename, delete, modify properties such as pricing, licensing comments and description, set PAR2 or MD5/SHA-[1,256,512] integrity functionalities, check for duplicates in the repository, etc.)
- Register/delete users and groups (except itself)
- Manage groups by assigning existing users and by configuring a set of properties
- Assign/revoke allowed applications to users/groups
- Assign/revoke licenses to users/groups
- Configure access methods (via mobile application or via web), security (HTTP or HTTPS) and type of login (based on device ID, based on credentials or hybrid)
- Manage system logs and configure elements to be logged (application downloads, system accesses, etc.), including billing reports

A normal user, by default can:

- Modify its profile (password if any, user information)
- Buy / download an application
- Comment and rate (optional) the downloaded applications

As mentioned at the beginning of the section, an API to develop low scale application stores was also provided. Depending on the needs of the enterprise, the resulting stores provide more or less functionality, and the modeled objects (*users*, *groups* and *applications*) have more or less attributes (where *applications* concept can be extended to *content*, including books, multimedia or other kind of products). Also, it is possible to choose the data source for the stored information, using XML files instead of databases, or using any other database instead of SQLite.

Because the API is written in Java, the final application can be deployed on various operating systems supporting JVM (Java Virtual Machine). Also, a wide range of Java compatible technologies can be used to provide enhanced functionality. I.e.: in case of developing a web application as frontend for the store, depending on the requisites, several webservers can be chosen (*Apache Tomcat*, *Bea Weblogic* or *IBM WebSphere* among others). These web servers can provide enhanced features, such as support for SSL connections using certificates and load balance. Features that ease the burden of programming the personal application store by not having to be implemented again.

In Table 1 authors present a summary of the main characteristics provided by the system, related to the wanted requirements.

	Versatility	**Portability**	**Security**	**Scalability**	**Reliability**	**Flexibility**	**Efficiency**

Use of Java programming language		**x**	**x**	**x**		**x**	**x**
User management functions	**x**		**x**				
Content management functions	**x**			**x**	**x**		
Possibility of external webserver usage	**x**	**x**	**x**	**x**	**x**	**x**	**x**
Possibility of different data source usage	**x**	**x**	**x**	**x**	**x**	**x**	**x**
Logs and billing reports			**x**		**x**		

Table 1 – Features vs. requirements

3 Application scenarios

Figure 1 shows an example of low scale enterprise app store architecture in which Android OS applications are distributed. The presented app store runs on JRE (Java Runtime Edition) without having into account the server operating system. Also, the data model is independent of the operating system. The file system in which database and mobile applications are present can be local or remote (they could be in another system for security or scalability reasons). In the proposed example, a web application is present. This web application uses several functionalities from the app core (based on ModEAS API) and it runs directly on a Java based web server (i.e. *Apache Tomcat*), which provides secure connections with the clients using SSL and certificates. The web application consists in a frontend for the app store in different flavors (desktop and mobile versions). On the other hand, there exist a local GUI to manage the app store and a web application which provide similar functionality (if administrator is not present in the enterprise or has no access to the server, is possible to manage the app store via web). The local GUI is also written in Java, so it is multiplatform and it can be integrated seamlessly with the app core.

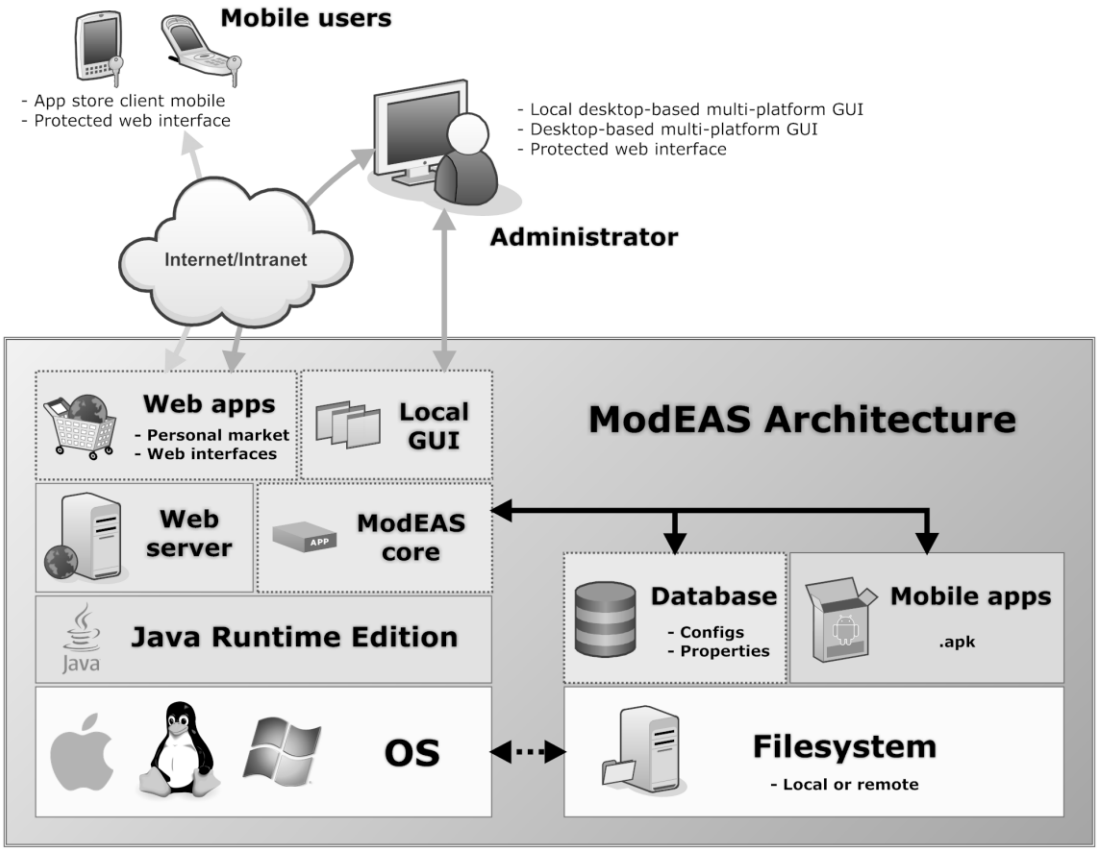

Figure 1 – Personal application store architecture

As previously mentioned, not only application markets can be developed with the provided API. Figure 2 shows an example of application (or *content*) repository management, following architectural recommendations presented in (V. Stantchev & Franke, 2010)(Vladimir Stantchev, 2010). This can be considered as an important functionality of the app store, but in this case, just this functionality is implemented. This application can be run in a single PC, and the application is just composed by a SQLite database, a set of applications stored in the local file system and classified in the application core. Also, three different management interfaces can be implemented:

- Local desktop-based Java application
- Remote desktop-based multi-platform application
- Protected web interface

This application can be used by individuals or by enterprises to classify their own applications or general content, taking advantage of some API functionalities, such as duplicates detection or parity-recovery information (which can prevent-recover damaged content).

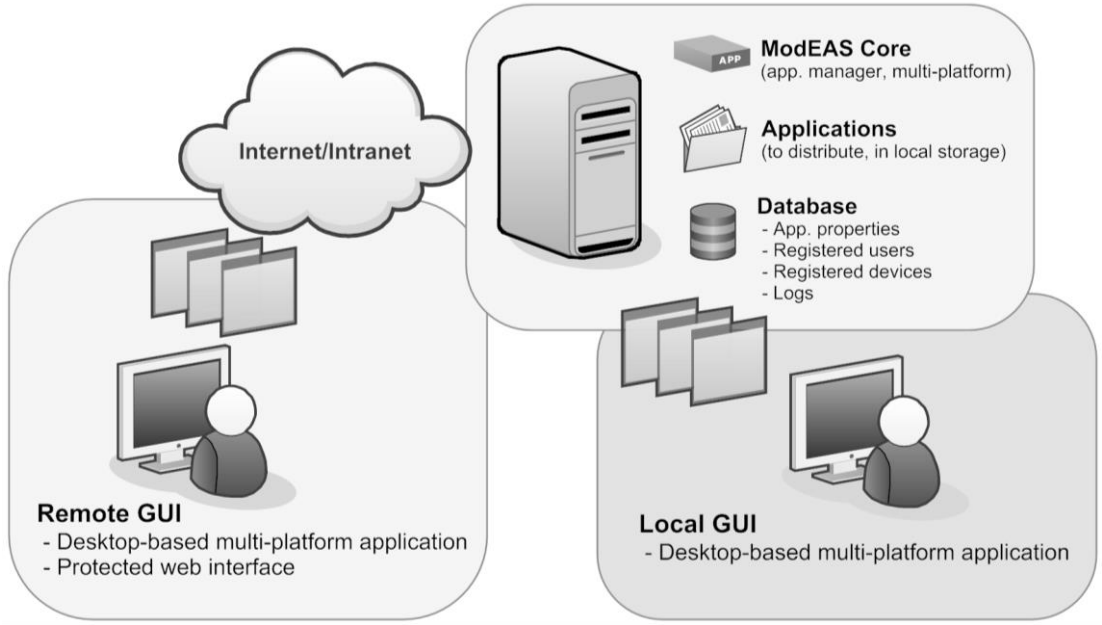

Figure 2 – Scenario 1: application repository management

Figure 3 extends the application presented in Figure 2. This scenario represents a personal application store, which could be implemented in a low scale enterprise. In this case, there is a web application, which could be the same as the one in Figure 1, and also a mobile client application. Final users of the system can access the app store via their portable devices, and they can do it through the Internet or through the enterprise WLAN. Security policies can be configured to be more restrictive if the detected device connection comes from the Internet. In this case, apart from a valid *device_id*, it will be necessary to provide additional login information (such *user_id/password*). If the mobile device connects through the local WLAN, it is just necessary provide a valid *device_id*. This makes the application more comfortable to use but without neglecting safety.

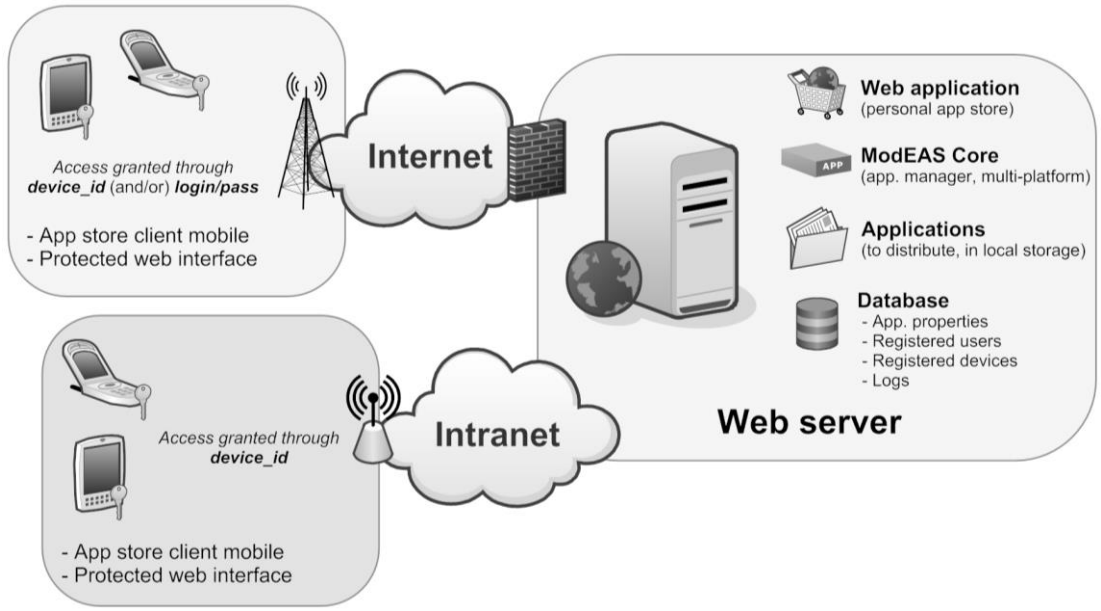

Figure 3 – Scenario 2: personal application store

3 Validation

Using the API, authors developed a simple JSP web application store to be tested. This application includes a basic user support using a SQLite database, a web interface with a basic *device_id* based authentication scheme, but without HTTPS configuration present in the webserver. This web application was deployed in Apache Tomcat 7, under Java 1.7 u17 64-bit in a Windows 7 Pro environment. The test environment is based on Apache JMeter. It is very important to consider that the given results are approximate and may vary depending on the environment in which test runs, since a change in the operating system, file system type, application data source or web server can alter greatly this results.

The test scenario consists in three probes. The first one has ten virtual concurrent users in ten iterations. The second one has twenty virtual concurrent users in five iterations, and finally the last execution has fifty virtual concurrent users in two iterations, which makes one hundred executions of the test script for each probe.

Probe	Average memory	Mean response time
10 users, 10 iterations	880 MB	73 ms
20 users, 5 iterations	900 MB	157 ms
50 users, 2 iterations	900 MB	224 ms

Table 2 – Performance tests results

Results show that the framework presents remarkable performance and trustable throughput to support a custom enterprise app store capable of running in a low-end computer.

4 Conclusions and Future Work

The Bring Your Own Device (BYOD) revolution has swept enterprises of all kinds and organisations as diverse as law firms and manufacturers, employees are buying their own mobile devices such as smartphones and tablets and using them for work (Romer, 2014). Some of the challenging issues detected in the literature come from apps installed in the phones. Taking this into account, this paper presents an effort devoted to design and implement enterprise app stores using ModEAS. Tests show that the approach is feasible in terms of performance and reliability.

The work presented in this paper proposes three types of initiatives which should be explored in future research. In the first place, it is intended to deploy social recommendation mechanisms for the apps stored in ModEAS. In the second place, it is aimed to develop upload and assessment features for new applications. Finally, and regarding security, it is planned to implement user trust instruments including certificates signed by trusted certification authorities and visual environmental security checks.

Acknowledgements

This paper has been partially supported by the European Commission (programme LifeLong Learning – action Leonardo da Vinci – Transfer of Innovation) through project "ECQA Certified Social Media Networker Skills" (2011-1-ES1_LEO05-35930), by the Spanish Ministry of Economy and Competitiveness through INNPACTO project Post-Via 2.0 (IPT-2011-0973-410000) and by the German Federal Ministry of Economics and Technology through project "PrevenTAB" (KF3144902DB3).

References

Colombo-Mendoza, L. O., Alor-Hernández, G., Rodríguez-González, A., & Colomo-Palacios, R. (2013). Alexandria: a visual tool for generating multi-device rich internet applications. *Journal of Web Engineering*, *12*(3-4), 317–359.

Colomo-Palacios, R., Fernandes, E., Sabbagh, M., & de Amescua Seco, A. (2012). Human and intellectual capital management in the cloud: software vendor perspective. *Journal of Universal Computer Science*, *18*(11), 1544–1557.

Costello, T., & Prohaska, B. (2013). 2013 Trends and Strategies. *IT Professional*, *15*(1), 64–64. doi:10.1109/MITP.2013.5

Cuadrado, F., & Dueñas, J. C. (2012). Mobile application stores: success factors, existing approaches, and future developments. *IEEE Communications Magazine*, *50*(11), 160–167. doi:10.1109/MCOM.2012.6353696

Finley, I., Redman, P., Prentice, B., & Buchanan, S. (2013). *Enterprise App Stores Can Increase the ROI of the App Portfolio*. Gartner Inc,.

FIPS 180-1. Secure hash standard. (1996). *NIST, US Department of Commerce, Washington D.C., Springer-Verlag*. Retrieved from http://www.itl.nist.gov/fipspubs/fip180-1.htm

Gartner. (2012). *Gartner Identifies the Top 10 Strategic Technology Trends for 2013*. ORLANDO, Fla: Gartner, Inc. Retrieved from http://www.gartner.com/newsroom/id/2209615

Gedda, R. (2012). Analyst view: 'Appy days. *CIO*, (Mar/Apr 2012), 18.

Hsieh, J.-K., & Hsieh, Y.-C. (2013). Appealing to Internet-based freelance developers in smartphone application marketplaces. *International Journal of Information Management*, *33*(2), 308–317. doi:10.1016/j.ijinfomgt.2012.11.010

Ismail, D. K. B., Douillard, C., & Kerouédan, S. (2013). A survey of three-dimensional turbo codes and recent performance enhancements. *EURASIP Journal on Wireless Communications and Networking*, *2013*(1), 1–13.

Kim, J., Park, Y., Kim, C., & Lee, H. (2014). Mobile application service networks: Apple's App Store. *Service Business*, *8*(1), 1–27. doi:10.1007/s11628-013-0184-z

Miller, K. W., Voas, J., & Hurlburt, G. F. (2012). BYOD: Security and Privacy Considerations. *IT Professional*, *14*(5), 53–55.

Morrow, B. (2012). BYOD security challenges: control and protect your most sensitive data. *Network Security*, *2012*(12), 5–8. doi:10.1016/S1353-4858(12)70111-3

Nahas, M., Clements, P., Nettle, P., & Gallagher, R. (2003). Parity Volume Set Specification 2.0. Retrieved from http://www.par2.net/par2spec.php

Patterson, D. A., Gibson, G., & Katz, R. H. (1988). A case for redundant arrays of inexpensive disks (RAID). *SIGMOD Rec.*, *17*(3), 109–116. doi:10.1145/971701.50214

Rivest, R. (1992). The MD5 message-digest algorithm. *Request for Comments, Internet Activities Board, Internet Privacy Task Force*. Retrieved from http://tools.ietf.org/html/rfc1321

Romer, H. (2014). Best practices for BYOD security. *Computer Fraud & Security*, *2014*(1), 13–15. doi:10.1016/S1361-3723(14)70007-7

Soto-Acosta, P., Martinez-Conesa, I., & Colomo-Palacios, R. (2010). An empirical analysis of the relationship between IT training sources and IT value. *Information Systems Management*, *27*(3), 274–283.

Stantchev, V., & Franke, M. R. (2010). Knowledge and learning aspects of project portfolio management. *International Journal of Knowledge and Learning*, *6*(2), 114–129.

Stantchev, Vladimir. (2010). Social computing for the public facility management in Berlin. *International Journal of Social and Humanistic Computing*, *1*(3), 261–272.

Thomson, G. (2012). BYOD: enabling the chaos. *Network Security*, *2012*(2), 5–8. doi:10.1016/S1353-4858(12)70013-2

Walters, R. (2013). Bringing IT out of the shadows. *Network Security*, *2013*(4), 5–11. doi:10.1016/S1353-4858(13)70049-7

BIOGRAPHYES

Lisardo Prieto-González is a researcher at SRH Hochschule Berlin, Germany and PhD student at the Computer Science Department of the Universidad Carlos III de Madrid, Spain. Also, he worked as

researcher and Teaching Assistant for three years in Telematics Department in this later institution. His research interests include complex data architectures, applied artificial intelligence, applied information systems and gamification.

Ricardo Colomo-Palacios, Full Professor at the Computer Science Department of the Østfold University College, Norway. Formerly he worked at Universidad Carlos III de Madrid, Spain. His research interests include applied research in Information Systems, software project management, people in software projects and social and Semantic Web. He received his PhD in Computer Science from the Universidad Politécnica of Madrid (2005). He also holds a MBA from the Instituto de Empresa (2002). He has been working as Software Engineer, Project Manager and Software Engineering Consultant in several companies including Spanish IT leader INDRA. He is also an Editorial Board Member and Associate Editor for several international journals and conferences and Editor in Chief of International Journal of Human Capital and Information Technology Professionals.

Beatriz Puerta is a researcher at the Computer Science Department of the Universidad Carlos III de Madrid, Spain. She studied Computer Science Bachelor in this institution. Her research interests include information security, artificial intelligence and mobile information systems development.

Prof. Dr. Gerrit Tamm is professor of information systems at SRH University Berlin and CEO of Asperado GmbH. After studying industrial engineering and business administration at the Technische Universität Berlin and at the University of California Berkeley he was member of the graduate school "distributed information systems" where he received his doctoral degree on "Web-based services: supply, demand and matching" from the Humboldt-Universität zu Berlin. He worked for 1 year as a pos doc at University St.Gallen, Switzerland. He was the executive director of the BMBF-funded Berlin Research Center on Internet Economics "Internet and Value Chains - InterVal" and the executive director of the BMWi-funded Research Center of Collaboration and RFID "Ko-RFID" in Berlin. He is founder and executive director of the electronic business forum, absolvent.de and Asperado GmbH. Prof. Dr. Tamm is member of the advisory committee of eco - Verband der deutschen Internetwirtschaft e.V.