­­MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Cloud-based Backend as a Service for Building Mobile Applications

|  |  |
| --- | --- |
| Nhóm 7 | |
| Group member | Nguyễn Mạnh Hùng – SE61388 Nguyễn Hữu Lộc – SE61312  Phạm Bảo Toàn – SE61011  Vũ Văn Quyết – SE61071 |
| Supervisor | Kiều Trọng Khánh |
| Ext. Supervisor | N/A |
| Capstone Project code | CSBM |

-Ho Chi Minh, *11-05-2016*-

**Table of Contents**

[**A. Introduction** 7](#_Toc457154652)

[**1.** **Project Information** 7](#_Toc457154653)

[**2.** **Introduction** 7](#_Toc457154654)

[**3.** **Current Situation** 7](#_Toc457154655)

[**4.** **Problem Definition** 7](#_Toc457154656)

[**5.** **Proposed Solution** 8](#_Toc457154657)

[**5.1. Feature functions** 8](#_Toc457154658)

[**5.2. Advantages and Disadvantages** 8](#_Toc457154659)

[**6.** **Functional Requirements** 8](#_Toc457154660)

[**7.** **Role and Responsibility** 9](#_Toc457154661)

[**B. Software Project Management Plan** 10](#_Toc457154662)

[**1.** **Problem Definition** 10](#_Toc457154663)

[**1.1. Name of this Capstone Project** 10](#_Toc457154664)

[**1.2. Problem Abstract** 10](#_Toc457154665)

[**1.3. Project Overview** 10](#_Toc457154666)

[**2.** **Project organization** 13](#_Toc457154667)

[**2.1. Software Process Model** 13](#_Toc457154668)

[**2.2. Roles and responsibilities** 14](#_Toc457154669)

[**2.3. Tools and Techniques** 14](#_Toc457154670)

[**3.** **Project Management Plan** 15](#_Toc457154671)

[**3.1. Product Backlog** 15](#_Toc457154672)

[**3.2. Deliverables** 16](#_Toc457154673)

[**3.3. All Meeting Minutes** 16](#_Toc457154674)

[**4.** **Coding Convention** 16](#_Toc457154675)

[**C. Software Requirement Specification** 18](#_Toc457154676)

[**1.** **User Requirement Specification** 18](#_Toc457154677)

[**1.1. Guest Requirement** 18](#_Toc457154678)

[**1.2. User Requirement** 18](#_Toc457154679)

[**1.3. Admin Requirement** 18](#_Toc457154680)

[**1.4. Authorized User Requirement** 18](#_Toc457154681)

[**2.** **System Requirement Specification** 18](#_Toc457154682)

[**2.1. External Interface Requirement** 18](#_Toc457154683)

[**2.2.** **System Overview Use Case** 19](#_Toc457154684)

[**2.3. List of Use Case** 20](#_Toc457154685)

[**3.** **Software System Attribute** 38](#_Toc457154686)

[**3.1. Usability** 38](#_Toc457154687)

[**3.2. Reliability** 38](#_Toc457154688)

[**3.3. Availability** 38](#_Toc457154689)

[**3.4. Security** 38](#_Toc457154690)

[**3.5. Maintainability** 38](#_Toc457154691)

[**3.6. Portability** 38](#_Toc457154692)

[**3.7. Performance** 38](#_Toc457154693)

[**4.** **Conceptual Diagram** 38](#_Toc457154694)

[**D. Software Design Description** 38](#_Toc457154695)

[**1.** **Design Overview** 38](#_Toc457154696)

[**2.** **System Architectural Design** 38](#_Toc457154697)

[**2.1 Web application architecture description** 38](#_Toc457154698)

[**2.2 CSBM application architecture description** 38](#_Toc457154699)

[**2.3 Mobile framework architecture description** 38](#_Toc457154700)

[**3.** **Component Diagram** 38](#_Toc457154701)

[**4.** **Detail Description** 38](#_Toc457154702)

[**4.1 Class Diagram** 38](#_Toc457154703)

[**4.2 Class Diagram Explanation** 38](#_Toc457154704)

[**4.3 Interaction Diagram** 38](#_Toc457154705)

[**5.** **Interface** 38](#_Toc457154706)

[**5.1 Web Service** 38](#_Toc457154707)

[**5.2 Mobile Framework APIs** 38](#_Toc457154708)

[**4.1.1.1** **CSBMClientConfiguration** 39](#_Toc457154709)

[2.1 Connecting to CSBM 39](#_Toc457154710)

[2.2 Enabling Local Datastore 39](#_Toc457154711)

[2.3 Enabling Extensions Data Sharing 39](#_Toc457154712)

[2.4 Other Properties 39](#_Toc457154713)

[2.5 Creating a Configuration 39](#_Toc457154714)

[**4.1.1.2** **BE** 39](#_Toc457154715)

[3.1 Accessing the Current User 39](#_Toc457154716)

[3.2 Creating a New User 39](#_Toc457154717)

[3.3 Logging In 39](#_Toc457154718)

[3.4 Becoming a User 39](#_Toc457154719)

[3.5 Revocable Session 39](#_Toc457154720)

[3.6 Logging Out 39](#_Toc457154721)

[3.7 Requesting a Password Reset 39](#_Toc457154722)

[**4.1.1.3** **BEACL** 39](#_Toc457154723)

[4.1 Creating an ACL 39](#_Toc457154724)

[4.2 Controlling Public Access 39](#_Toc457154725)

[4.3 Controlling Access Per-User 39](#_Toc457154726)

[4.4 Controlling Access Per-Role 39](#_Toc457154727)

[4.5 Setting Access Defaults 39](#_Toc457154728)

[**4.1.1.4** **BEObject** 39](#_Toc457154729)

[5.1 Creating a BEObject 39](#_Toc457154730)

[5.2 Managing Object Properties 39](#_Toc457154731)

[5.3 Accessor 39](#_Toc457154732)

[5.4 Array Accessors 39](#_Toc457154733)

[5.5 Increment 39](#_Toc457154734)

[5.6 Saving Objects 39](#_Toc457154735)

[5.7 Saving Many Objects 39](#_Toc457154736)

[5.8 Deleting Many Objects 39](#_Toc457154737)

[5.9 Getting an Object 39](#_Toc457154738)

[5.10 Getting Many Objects 39](#_Toc457154739)

[5.11 Fetching From Local Datastore 39](#_Toc457154740)

[5.12 Deleting an Object 39](#_Toc457154741)

[5.13 Pinning 39](#_Toc457154742)

[5.14 Pinning Many Objects 39](#_Toc457154743)

[5.15 Unpinning 39](#_Toc457154744)

[5.16 Unpinning Many Objects 39](#_Toc457154745)

[**4.1.1.5** **BEFile** 39](#_Toc457154746)

[6.1 Creating a BEFile 39](#_Toc457154747)

[6.2 File Properties 39](#_Toc457154748)

[6.3 Storing Data with CSBM 39](#_Toc457154749)

[6.4 Getting Data from CSBM 39](#_Toc457154750)

[6.5 Interrupting a Transfer 39](#_Toc457154751)

[**4.1.1.6** **BEQuery** 39](#_Toc457154752)

[7.1 Creating a Query for a Class 39](#_Toc457154753)

[7.2 Adding Basic Constraints 39](#_Toc457154754)

[7.3 Adding Location Constraints 39](#_Toc457154755)

[7.4 Adding String Constraints 39](#_Toc457154756)

[7.5 Adding Subqueries 39](#_Toc457154757)

[7.6 Sorting 39](#_Toc457154758)

[7.7 Getting Objects by ID 39](#_Toc457154759)

[7.8 Getting all Matches for a Query 39](#_Toc457154760)

[7.9 Getting the First Mathc in Query 39](#_Toc457154761)

[7.10 Counting the Mathches in a Query 39](#_Toc457154762)

[7.11 Cancelling a Query 39](#_Toc457154763)

[7.12 Paginating Results 39](#_Toc457154764)

[7.13 Controlling Caching Behavior 39](#_Toc457154765)

[7.14 Query Source 39](#_Toc457154766)

[7.15 Advanced Settings 39](#_Toc457154767)

[**6.** **Database Design** 39](#_Toc457154768)

[**6.1 Entity Relationship Diagram** 39](#_Toc457154769)

[**6.2 Data Dictionary** 39](#_Toc457154770)

[**7.** **Algorithms** 39](#_Toc457154771)

**Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| Name | Definition |
| BaaS | Back-end as a service |

# **A. Introduction**

## **Project Information**

- Project name: **Cloud-based Backend as a Service for Building Mobile**

**Applications.**

- Project code: **CSBM.**

- Project type: **Web Application and Mobile Framework.**

- Start date: **11-05-2016.**

- End date: **21-8-2016.**

## **Introduction**

In this document, we introduce a solution for mobile developers. Developing mobile applications is more and more popular and cloud computing also extremely develops. The combination is very complex to build the effective application on smartphone. Based on our researches and analysis, we proposed a cloud-based backend as a service for mobile developers.

We provide a service that includes website application and mobile framework, which helps mobile developers to develop their application quickly, easily to change, and flexibility in modification. Somethings supported the developer building their application without worrying about the backend services.

## **Current Situation**

When developing mobile applications, developers need to concern too much about back-end such as: creating service support push notification, creating and managing database mobile… Most modern mobile applications store data and interact with other services on the internet. User accounts, shared content, documents and purchases; these things all need to be stored somewhere else, and cloud is a good solution. There is a cloud-based backend as a service of Facebook named “Parse” (https://www.parse.com), however the “Parse” is shutdown in first month 2016…

## **Problem Definition**

Others backend as a service:

* Advantages:
* Parse, Firebase… are well-known and already had much users.
* Disadvantages:
  + Parse (http://parse.com/) is shutdown in first month 2017.
  + CloudKit (<https://developer.apple.com/icloud/>) is limited to iOS and require user to use iCloud.
  + Firebase (<https://www.firebase.com/>) hasn’t push notification

Common mobile development:

* Advantages:
* Taking control everything from mobile layout to back-end component.
* Disadvantages:
* Mobile developers need to have the knowledge about web service or wait provided services from web service developer.
* Mobile developers hard to focus on creating user experience.
* Mobile application development is waste of time on building API.

## **Proposed Solution**

Our proposed solution is to develop the Cloud-based Backend as a Service named “CSBM” to support mobile developer manage application data via our provided framework and only focus on creating extraordinary user experiences. We’ll take care of the rest.

### **5.1. Feature functions**

* Web application: design for developer to manage data and configuration of their mobile application.
* Server: parse data from client side to store and sync data with our NoSQL cloud database.
* Mobile framework: provide framework for mobile developer to interact with our database.

### **5.2. Advantages and Disadvantages**

* **Advantages:**
  + Real-time database: data is stored in cloud database and synced to all connected clients in real-time.
  + Push notification: auto push notification to all connected client after data is changed.
  + Mobile framework for developers to interact with their data on server database.
* **Disadvantages:**
  + Security: users only manage their data while the security is based on the service provided.
  + Application downtime: users can’t control downtime when CSBM server has problem.

## **Functional Requirements**

* **Server component**
  + Parse data from client dashboard to system storage.
  + Notify to connected client.
  + Sync database to server automatically.
* **Web component**
  + Manage applications.
  + Manage data.
* **Mobile framework** 
  + Provide framework for mobile.

## **Role and Responsibility**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full Name | Role | Position | Contact |
| 1 | Kiều Trọng Khánh | Project Manager | Supervisor | khanhkt@fpt.edu.vn |
| 2 | Nguyễn Mạnh Hùng | Developer | Leader | hungnmse61388@fpt.edu.vn |
| 3 | Nguyễn Hữu Lộc | Developer | Member | locnhse61312@fpt.edu.vn |
| 4 | Vũ Văn Quyết | Developer | Member | quyetvv61071@fpt.edu.vn |
| 5 | Phạm Bảo Toàn | Developer | Member | toanpbse61011@fpt.edu.vn |

*Table 1: Roles and Responsibilities*

# **B. Software Project Management Plan**

## **Problem Definition**

### **1.1. Name of this Capstone Project**

- Official name: Cloud-based Backend as a Service for Building Mobile Applications.

- Vietnamese name: Dịch vụ đám mây hỗ trợ developer xây dựng mobile application.

- Abbreviation: CSBM.

### **1.2. Problem Abstract**

At the moment, especially in Viet Nam, not much developer use BaaS (Backend as a Service). The development of mobile application include building back-end so it requires mobile developer have knowledge about coding web api or they may wait for web api from others developers. This situation makes mobile developers can’t focus on creating good user experiences, and then the effective of build a mobile application is always not best liked developer hope.

Besides, currently there are Parse (<http://parse.com/>), Cloud-Kit (<https://developer.apple.com/icloud/>), Firebase (<https://www.firebase.com/)>,... But they have not provided developers a good way to build a mobile application. Cloud-Kit is limited to iOS and require user to use iCloud. Firebase hasn’t push notification. Parse is stop service provider in first month 2016 and is shutdown in first month 2017, but many applications are there. If Parse shutdown, developers who are using Parse will have difficulty in finding another BaaS like Parse.

For the goal that improving mobile development, we provide a service which base on Parse Server, to reduce the time consuming in mobile development and mobile developers can focus more on user interface and user experiences. By using Parse Server, we develop a website to create new applications, and a service to sharing data between user’s different devices. The service also haves push notifications, if either of those are important to developer applications, our service will likely save develop time.

### **1.3. Project Overview**

#### **1.3.1. Current Situation**

Below are the problems encountered in this project:

* + **Disadvantages:**
    - Parse source code was developed with NodeJs that nobody in team has much knowledge about.
    - Parse source code not public function to allow users create new application.
    - Parse server now can only run one application. If developers have many application they will need many Parse server to run all of them.
    - Team hasn’t much knowledge about how to code and build a mobile framework.
  + **Advantages:**
    - The server is ready to run based on Parse source code.

#### **1.3.2. The Proposed System**

According to technology researches, we found out that Parse server of Facebook can help us resolving the problem of reducing time consuming in mobile development and also help developers who are using Parse now can move to and continue develop application.

To resolve the problem that Parse uses NodeJs as main language for the server, we can use NodeJs Tutorial from <http://www.tutorialspoint.com/nodejs/> for understanding how NodeJs work. We also find the necessary help from NodeJs development forum to solve problems.

We must to know what dashboard send request to server to create a new application to write a function allow developer can create new application from our service dashboard. According to technology researches, we decided AngularJs will be the framework for dashboard.

We assign responsibility in horizontal, it means everyone will do a whole big think and separated from the others. This decision based on that everything about this project is really new for all of the member so let each one focus on one problem is the best way.

##### **1.3.2.1. Website (Dashboard)**

Main web application is a place for developer to create a new application or register account to use main function of the service website. Developer also can use this website to management the data of their application.

* For developer
  + - Create a new application: Developer can create a new application.
    - Manage data: Create/Edit/Remove developer data in application.
  + For guest
    - Register: User can register new account to use the service.

##### **1.3.2.2. Mobile Framework**

This framework is use for Developer’s mobile Application, which provide APIs to control mobile application data. Mobile developers can control their data really easily after download framework which appropriate to their language (Android or iOS) and apply to their application.

#### **1.3.3. Boundaries of the System**

* This service is built basing on Parse server of Facebook. Our main target is improving the mobile development and help developers, whose applications is using Parse now, have an appropriate place to migrate to.
* The complete product includes:
  + A web application that allow:
    - Register account to use main function of the system
    - Create a new application
    - Manage data
  + Mobile Framework that allow:
    - Provide API

#### **1.3.4. Future Plans**

Current service only support for mobile. In future we will:

* + - Provide framework for website developers.

#### **1.3.5. Development Environment**

##### **1.3.5.1. Hardware requirements**

* + For server

|  |  |  |
| --- | --- | --- |
| Hardware | Minimum Requirements | Recommended |
| Internet Connection | 512Kbps | 4 Mbps |
| Operating System | Ubuntu Server 12 LTS | Ubuntu Server 14.04.2 LTS |
| Computer Processor | Intel® CORE i3 Quad core 2.1 GHz | Intel® CORE i7 Quad core 2.4 GHz |
| Computer Memory | 1GB of RAM | 4GB of RAM or more |

*Table 2: Hardware Requirement for server*

* + For website development

|  |  |  |
| --- | --- | --- |
| Hardware | Minimum Requirements | Recommended |
| Internet Connection | 512Kbps | 4 Mbps |
| Operating System | Ubuntu Server 12 LTS | Ubuntu Server 14.04.2 LTS |
| Computer Processor | Intel® CORE i3 Quad core 2.1 GHz | Intel® CORE i7 Quad core 2.4 GHz |
| Computer Memory | 1GB of RAM | 4GB of RAM or more |

*Table 3: Hardware Requirement for website development*

* + For mobile framework development

|  |  |  |
| --- | --- | --- |
| Hardware | Minimum Requirements | Recommended |
| Internet Connection | 512Kbps | 4 Mbps |
| Operating System | Ubuntu Server 12 LTS | Ubuntu Server 14.04.2 LTS |
| Computer Processor | Intel® CORE i3 Quad core 2.1 GHz | Intel® CORE i7 Quad core 2.4 GHz |
| Computer Memory | 1GB of RAM | 4GB of RAM or more |

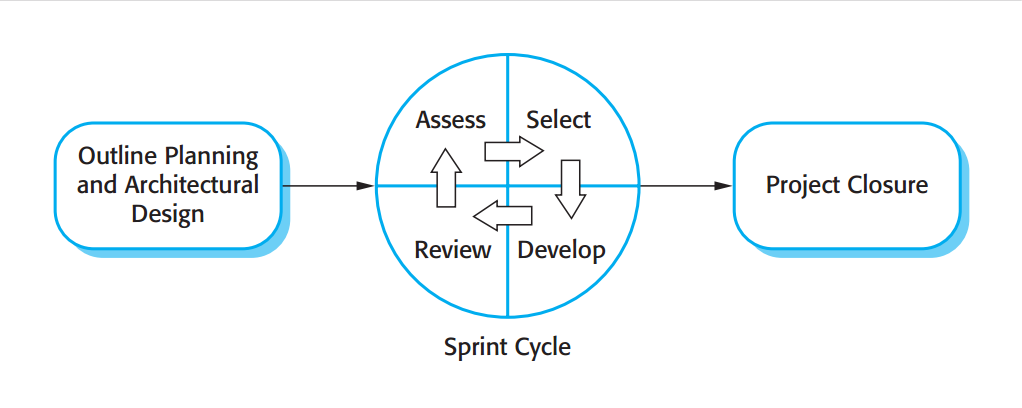
*Table 4: Hardware Requirement for mobile framework development*

## **Project organization**

### **2.1. Software Process Model**

The project is developed under Scrum model. We choose this model because the following reasons:

* + This project is about Parse server, NodeJs and building mobile framework these are really new to all of member. That why our team need focus on research, study and try our best to implement a core flow of this project early.
  + We need to provide many demo for each requirement to make sure that our team can complete this project.
  + This project is about building framework so it’s really hard to finish the document first. We need coding through some demo to understand more about the operation mechanism of Parse server and figure out how to build framework effectively.



*Figure 1: Scrum model (Source: Software Engineering, 9th Edition, Chapter 3, Figure 3.8)*

### **2.2. Roles and responsibilities**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| 1 | Mr. Kiều Trọng Khánh | Scrum Product Owner – Technical Expert | * Specify user requirement * Specifying the business * Control the development process * Give advices on techniques, solutions and business analysis support |
| 2 | Nguyễn Mạnh Hùng | Scrum Master, Business Analyst, Developer, Tester | * Managing process * Researching solutions and techniques * Assigning task for members * Study Parse server for building “Create new app”. |
| 3 | Nguyễn Hữu Lộc | Team Member, Business Analyst, Developer, Tester | * Researching solutions and techniques * Build framework for iOS developers. |
| 4 | Vũ Văn Quyết | Team Member, Business Analyst, Developer, Tester | * Researching solutions and techniques * Build framework for Android developers. |
| 5 | Phạm Bảo Toàn | Team Member, Business Analyst, Developer, Tester | * Researching solutions and techniques * Build Dashboard |

*Table 5: Roles and Responsibilities Details*

### **2.3. Tools and Techniques**

|  |  |
| --- | --- |
| **Tools** | **Name / version** |
| Html, Javascript Editor | Sublime Text V3 |
| iOS IDE | Xcode 7.2 |
| Android IDE | Android Studio 2.1 |
| DBMS | Robomongo 0.9.0 RC8 |
| Source control | GitHub |
| Document | Microsoft Word 2016 |

*Table 6: Tools used for this project*

|  |  |
| --- | --- |
| **Techniques** | **Name / version** |
| Backend | NodeJs 6.2.0 |
| Frontend | AngularJs 1.5.0 |
| iOS Framework | Objective-C 2.0 |
| Android Framework | Java |
| DBMS | MongoDB 3.2.6 |

*Table 7: Techniques used for this project*

## **Project Management Plan**

### **3.1. Product Backlog**

|  |  |  |
| --- | --- | --- |
| **Id** | **User Story** | **Priority** |
| 1 | Create new application | High |
| 2 | Manage applications | High |
| 3 | Manage user account for an application | High |
| 4 | Mange user role for an application | Medium |
| 5 | Multiple application on one Parse server | High |
| 6 | Push notification | Low |
| 7 | Framework for Android | High |
| 8 | Framework for iOS | High |
| 9 | Application search | Low |
| 10 | Login | High |
| 11 | Logout | Low |
| 12 | Auto generate ApplicationId | Medium |
| 13 | Analytics | Low |

### **3.2. Deliverables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Deliverable** | **Delivery date** | **Delivery Location** | **Note** |
| 1 | Introduction Document | 12/5/2016 | FPT-CMS | Report No.1 |
| 2 | Software Project Management Plan | 16/5/2016 | FPT-CMS | Report No.2 |
| 3 | Software Requirement Specification | 30/5/2016 | FPT-CMS | Report No.3 |
| 4 | Application Demo 1 | 16/6/2016 | FPT-University | Core flow demo |
| 5 | Application Demo 2 | 21/7/2016 | FPT-University | Features demo |
| 6 | Design 1 | 21/7/2016 | FPT-CMS | Report No.4 |
| 7 | Application Demo 3 | 7/8/2016 | FPT-University | Semi-final application demo |
| 8 | Design 2 | 7/8/2016 | FPT-CMS | Report No.5 |
| 9 | Completed Document | 14/8/2016 | FPT-CMS | Report No.6 |

### **3.3. All Meeting Minutes**

All meeting minutes here: <https://github.com/hungnm2904/CSBM/tree/report/Meeting%20Minutes>.

## **Coding Convention**

Summary:

* Naming Convention:
  + Variable names should be short yet meaningful. The choice of a variable name should be designed to indicate to the casual observer the intent of its use.
  + Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
  + All names start with a letter.
  + Variable and function names written as **camelCase**.
* Declarations Convention:
  + One declaration per line is recommended since it encourages commenting.

Using Java Code Convention from:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

<http://www.w3schools.com/js/js_conventions.asp>

<https://github.com/mgechev/angularjs-style-guide>

Programming with Objective-C summary:

* Naming Convention:
  + Class names must be unique across an entire app.
  + Method names should be expressive and unique within a Class.
  + Local Variables must be unique within The same scope.
  + Accessor method names Must follow conventions.
  + Object creation method names must follow conventions.

Using Object Convention from:

<https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Conventions/Conventions.html#//apple_ref/doc/uid/TP40011210-CH10-SW1>

# **C. Software Requirement Specification**

## **User Requirement Specification**

### **1.1. Guest Requirement**

Guest is a person who does not have access the system. Guest can use some functions in the system. To use all functions, guest must login. These are some functions guest can use:

* Login
* Register

### **1.2. User Requirement**

User is guest who already login to the system by his/her account and uses service of system. The customer can use some following functions:

* Manage application: Create New Application, Delete Application, Active Application.
* Manage class: Create Class, Update Class, Delete Class.
* Query data

### **1.3. Admin Requirement**

Person who manages account. Administrator can use some following functions:

* Change User status.
* Change Application status.

### **1.4. Authorized User Requirement**

Authorized User is the person who has accessed the system, besides the functions that user can use base on their role, authenticated user also can use the following function:

* Logout.

## **System Requirement Specification**

### **2.1. External Interface Requirement**

#### **2.1.1. User interface**

* The user interface use English language in web application and mobile framework.
* The user interface for website display best on 1024x768-screen size.

#### **2.1.2. Hardware interface**

* Computer:
  + OS: Ubuntu 14 LTS
  + CPU: Intel® CORE i3 Quad core 2.1 GHz
  + RAM: 1GB
* Mobile:

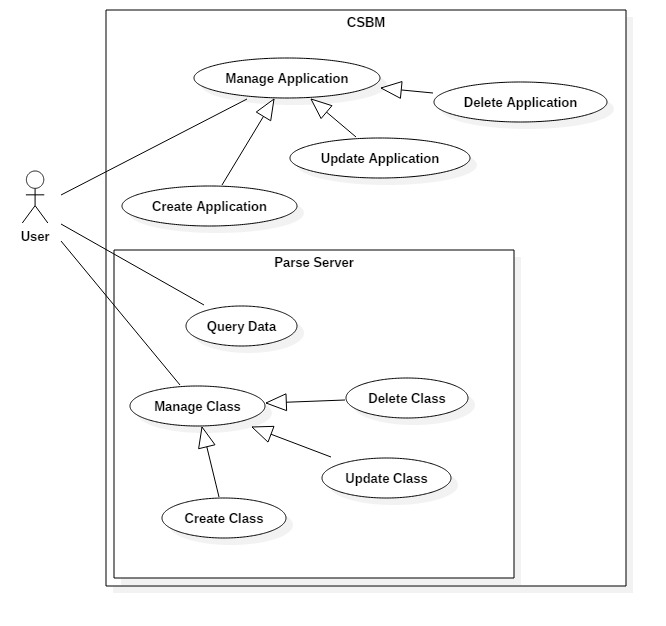
#### **2.1.3. Software Interface**

* Web application: work with Chrome (v47 or above), Internet Explorer (v10 or above), Firefox (v43 or above)
* Mobile framework: Android operation system (v 4.4 or above).

#### **2.1.4. Communication Protocol**

* Use Rest API for communication between the web browser and server.
* Use Rest API for communication between the mobile framework and server.

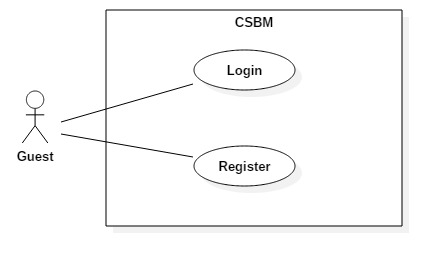
### **System Overview Use Case**



*Figure: System Overview Use Case*

### **2.3. List of Use Case**

#### **2.3.1. <Guest> Overview use case**



*Figure: <Guest> Overview Use Case*

##### **2.3.1.1. <Guest> Register**



*Figure: <Guest> Register*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC001** | | | |
| **Use Case No.** | 001 | **Use Case Version** | 1.0 |
| **Use Case Name** | Register | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * Guest   **Summary:**   * This use case allows guest register an account.   **Goal:**   * Create new account.   **Triggers:**   * Guest sends command to register.   **Preconditions:** N/A.  **Post Conditions:**   * **Success:** Guest will be logged in system with their roles. * **Fail:** Error message will be shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Guest sends command to request register | System requires information from Guest:   * Email: free text input, required, regex [^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@"+"[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$] * Full name: free text input, required, length (10-50) * Username: free text input, required, length (9-20) * Password: free text input, required, length (6-12) * Repeat password: free text input, required, length (6-12) * Phone: free number input, required, length (10-12) positive integer, value: [0,9] | | 2 | Guest inputs information |  | | 3 | Guest sends command to register  [Alternative 1]  [Alternative 2] | System shows login view  Account registered  [Exception 1]  [Exception 2]  [Exception 3]  [Exception 4] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Guest sends command to reset | System reset all field to blank | | 2 | Guest send command to back to login view | System shows login view  Account isn’t created |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Guest input already exist username | System show warning message “User name already exist” | | 2 | Guest input already exist email | System shows warning message “Email already exist”. | | 3 | Guest does not input required field. | System notices that guest need to input all these field:   * “Email”: System display warning message: “please fill out this field”. * “Full name”: System display warning message: “please fill out this field”. * “Username”: System display warning message: “please fill out this field”. * “Password”: System display warning message: “please fill out this field”. * “Repeat password”: System display warning message: “please fill out this field”. * “Phone”: System display warning message: “please fill out this field”. | | 4 | Guest input wrong some fields with requirement. | System notices that guest need to re-input all these field:   * “Email”: System display warning message: “Email invalid! ([me@example.com)](mailto:me@example.com))”. * “Email”: System display warning message: “Email must be 10 - 254 characters”. * “Full name”: System display warning message: “Full name must be 10 - 50 characters”. * “Username”: System display warning message: “Username must be 6 - 20 characters”. * “Password”: System display warning message: “Password must be 6 - 12 characters”. * “Repeat password”: System display warning message: “Repeat password does not match password”. * “Phone”: System display warning message: “Phone must be numbers”. |   **Relationships:** N/A  **Business Rules:**   * After registered, information of account will be stored in database of the system with role “User” and status is “active” | | | |

**Table: Register specification**

##### **2.3.1.2. <Guest> Login**

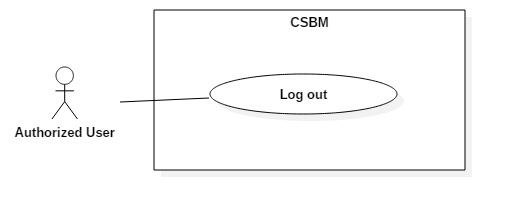


*Figure: <Guest> Login*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC002** | | | |
| **Use Case No.** | 002 | **Use Case Version** | 1.0 |
| **Use Case Name** | Login | | |
| **Author** |  | | |
| **Date** | 25/5/2016 | **Priority** | High |
| **Actor:**   * Guest   **Summary:**   * This use case allows authorized user to log in system.   **Goal:**   * Authentication and authorization.   **Triggers:**   * Guest send the login command.   **Precondition:**   * N/A   **Post Condition:**   * **Success:** Guest will be logged in system with their roles. * **Fail:** Error message will be shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Guest goes to login view. | System requires information:   * Username: textbox, required. * Password: textbox, required. | | 2 | Input field. |  | | 3 | Guest sends login command. | Guest will login system with their roles.  [Exception 1] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Guest sends login command. | Wrong username or password. Show error message: “Username or password invalid”. |   **Relationships:** N/A  **Business Rules:**   * After login successfully, guest will be redirected to specific view based on their role on the system: user or admin.   + If role is “User”, the system will display to User view.   + If role is “Admin”, the system will display to Admin view. | | | |

**Table: Login specification**

#### **2.3.2. <Authorized User> Overview use case**



*Figure: <Authorized User> Overview Use Case*

##### **2.3.2.1. <Authorized User> Logout**

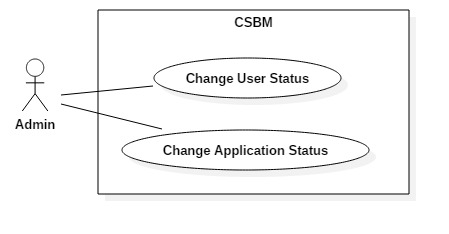


*Figure: <Authorized User> Logout*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC003** | | | |
| **Use Case No.** | 003 | **Use Case Version** | 1.0 |
| **Use Case Name** | Log out | | |
| **Author** |  | | |
| **Date** | 25/5/2016 | **Priority** |  |
| **Actor:**   * Authenticated user.   **Summary:**   * This use case allows Authenticated user logouts the system.   **Goal:**   * User can log out of system.   **Triggers:**   * User sends logout command.   **Precondition:** N/A.  **Post Condition:**   * **Success:** Display login view. * **Fail:** N/A   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User click button “Logout” in menu bar. | System clear session state if have, takes user out of system.  System display login view. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * If user logout while send request to system, all request will be delete. * If user is not available longer than 30 minutes, they will see the login view when they be back. | | | |

**Table: Logout specification**

#### **2.3.3. <Admin> Overview use case**



*Figure: <Admin> Overview Use Case*

##### **2.3.3.1. <Admin> Change Application Status**

*Figure: <Admin> Change Application Status*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC004** | | | |
| **Use Case No.** | 004 | **Use Case Version** | 1.0 |
| **Use Case Name** | Change Application Status | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * Admin.   **Summary:**   * This use case allows admin to update application status.   **Goal:**   * Update existed application.   **Triggers:**   * Admin click on “Change status” on an application in “Manage Application” page.   **Precondition:**   * Application exists in storage. * User must log into the system with role admin.   **Post Condition:**   * **Success:** Application status is updated into storage. * **Fail:** N/A.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Admin click on “Change status” button. | Confirmation dialog will be shown.  [Alternative 1,2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Ok” button. | Maker position will be saved into storage. | | 2 | Click on “Cancel” button. | Confirmation dialog will be canceled and no change is made. |   **Exceptions:** N/A.  **Relationships:** N/A.  **Business Rules:**   * If application’s status is changed to “Deactivate”, the application won’t be able to use and users won’t be able to access the application. * In owner view, users won’t be able to access the application has banner “Deactivate” with red text. * If application’s status is changed to “Activate”, users can access the application. | | | |

**Table: Change Application Status specification**

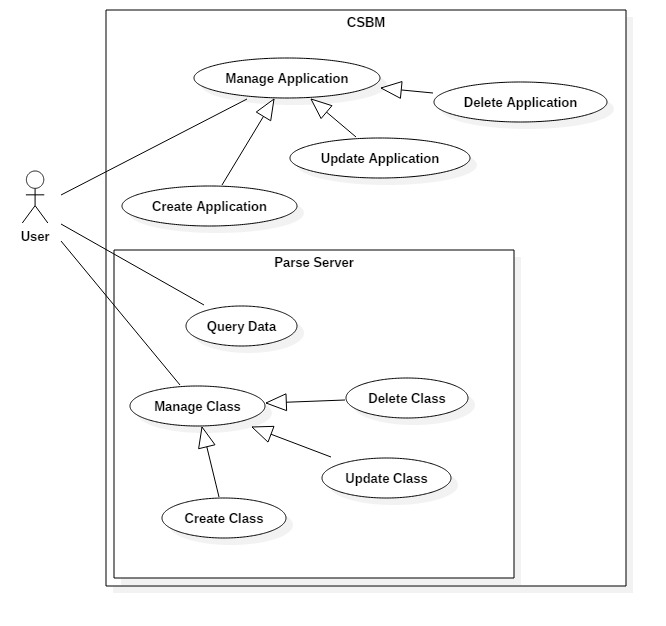
##### **2.3.3.2. <Admin> Change User Account Status**

*Figure: <Admin> Change User Account Status*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC004** | | | |
| **Use Case No.** | 004 | **Use Case Version** | 1.0 |
| **Use Case Name** | Change User Account Status | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * Admin.   **Summary:**   * This use case allows Admin to update user account.   **Goal:**   * Update exists user account.   **Triggers:**   * Admin click on “Change status” on an account in “Manage Application” page.   **Precondition:**   * Account exists in storage. * User must log into the system with role admin.   **Post Condition:**   * **Success:** Account status is updated into storage. * **Fail:** N/A.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Admin click on “Change status” button. | Confirmation dialog will be shown.  [Alternative 1,2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Ok” button. | Maker position will be saved into storage. | | 2 | Click on “Cancel” button. | Confirmation dialog will be canceled and no change is made. |   **Exceptions:** N/A.  **Relationships:** N/A.  **Business Rules:**   * If account’s status is changed to “Deactivate”, the account won’t be able to access the system. * Administrator can’t Deactivate himself. * If account’s status is changed to “Activate”, Guest can use this account to access the system. | | | |

**Table: Change User Status specification**

#### **2.3.4 <User> Overview use case**



*Figure: <User> Overview Use Case*

##### **2.3.4.1. <User> Create New Application**



*Figure: <User> Create New Application*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC008** | | | |
| **Use Case No.** | 008 | **Use Case Version** |  |
| **Use Case Name** | Create New Application | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * User.   **Summary:**   * This use case allows user to create new application.   **Goal:**   * New application will be stored in storage.   **Triggers:**   * User click “Create new application” button on “Manage Application” page.   **Precondition:**   * Actor login to system with role “User”.   **Post Condition:**   * **Success:** All new application’s data were saved successfully. * **Fail:** Error message will be shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User click “Add New Application” button. | System show dialog which contains information of:   * Application Name: textbox, min length 2, max length 10, required. * “Create”: button. | | 2 | Input field. |  | | 3 | Click on “Create” button. | Confirmation dialog will be shown  [Exception 1]  [Alternative 1, 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor action | System Response | | 1 | Click on “Ok” button | New application will be added in storage. | | 2 | Click on “Cancel” button | Confirm dialog will be canceled. A new application isn’t created. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Click on “Create” button | “Application Name” is empty or longer than 10 words. Show error message: “Application Name is limited from 2 to 10 characters”. |   **Relationships:** N/A.  **Business Rules:**   * New Application will be created with active status. * System must ensure has not duplicated application name. * System will notifies owner if the application is added fail. * When user add application from create new application form, system will automatic generate userId by Id’s user create application, random ObjectId, masterKey, generate created\_at and updated\_at by current time and save it to system, for example:  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | ObjectId | masterKey | created\_at | update\_at | name | userId | | 5759360ae77ed0391e143d45 | 38ccZfmdEJXE5axrKh1TTX7Qp3uAfLVjYZYi6mVQFNPi92hn | ISODate("2016-06-09T09:25:30.000Z") | ISODate("2016-06-09T09:25:30.000Z") | test | 5756aaaa6f9268590fa0867b | | | | |

**Table: Create New Application specification**

##### **2.3.4.2. <User> Delete Application**

*Figure: <User> Delete Application*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC009** | | | |
| **Use Case No.** | 009 | **Use Case Version** |  |
| **Use Case Name** | Delete Application | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows user to delete application.   **Goal:**   * Application will be removed from the storage.   **Triggers:**   * User click “Delete Application” button in “Manage Application” page.   **Precondition:**   * Actor login to system with role “User”. * Application exists in the system.   **Post Condition:**   * **Success:** Application will be deleted from storage. * **Fail:** Application is not deleted. Error message will be shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Application owner click on “Delete Application” button in “Manage Application” page. | System show dialog following information:   * “Application Name”: Dropdown list contain all application name, required. * “Delete”: button. | | 2 | Choose application name. |  | | 3 | Click on “Delete” button. | Confirmation dialog will be shown  [Exception 1]  [Alternative 1, 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor action | System Response | | 1 | Click on “Ok” button | Dialog is turn-off, Application is deleted. | | 2 | Click on “Cancel” button | Dialog is turn-off, Application is not deleted. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor action | System Response | | 1 | Click on “Ok” button | Application is not deleted. Show error message: “Delete application fail. Try again”. |   **Relationships:** N/A.  **Business Rules:**   * Application is still active. | | | |

**Table: Delete application specification**

##### **2.3.4.3. <User> Update Application**

*Figure: <User> Update Application*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC008** | | | |
| **Use Case No.** | 010 | **Use Case Version** |  |
| **Use Case Name** | Update Application | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * User.   **Summary:**   * This use case allows user to update existed application.   **Goal:**   * Update existed application.   **Triggers:**   * User click “Update Application” button in “Manage Application” page.   **Precondition:**   * Actor logged in system with role “User”. * Application exists in the system.   **Post Condition:**   * **Success:** Application will be updated with new information. * **Fail:** Application will not be updated. Error message will be shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User click “Update Application” button on “Manage Application” page. | System show dialog following information:   * “Application Name”: textbox, min length: 1, max length: 100, required. * “Update”: button. * “Close”: button. | | 2 | User fills in information application.  [Alternative 1] |  | | 3 | Click on “Update” button. | Confirmation dialog will be shown.  [Exception 1]  [Alternative 2, 3] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor action | System Response | | 1 | Click on “Close” button. | System close dialog and cancel update process. | | 2 | Click on “Update” button. | Application will be updated. | | 3 | Click on “Cancel” button. | Confirmation dialog will be canceled and no change is made. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Click on “Update” button. | “Application Name” is empty or longer than 100 words. Show error message: “Application Name is limited from 1 to 100 characters”. |   **Relationships:** N/A.  **Business Rules:**   * New information of application will be stored in storage. * System must ensure has not duplicated application name. | | | |

**Table: Update Application specification**

##### **2.3.4.4. <User> Update Class Field**

*Figure: <User> Update Class Field*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC008** | | | |
| **Use Case No.** | 011 | **Use Case Version** |  |
| **Use Case Name** | Update Class Field | | |
| **Author** |  | | |
| **Date** | 25/05/2016 | **Priority** |  |
| **Actor:**   * User.   **Summary:**   * This use case allows user to update existed class.   **Goal:**   * Update existed class.   **Triggers:**   * User click “Change Field” button in “Manage Application” page.   **Precondition:**   * Actor logged in system with role “User”. * Class Field exists in the system.   **Post Condition:**   * **Success:** Class field will be updated with new information. * **Fail:** Class field will not be updated. Error message will be shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User click “Change Field” button on “Manage Application” page. | System show dialog following information:   * “Field Name”: Dropdown list contain all class field name, required. * “New Field Name”: textbox, min length: 1, max length: 100, required. * “Update”: button. * “Close”: button. | | 2 | User fills in information application.  [Alternative 1] |  | | 3 | Click on “Update” button. | Confirmation dialog will be shown.  [Exception 1]  [Alternative 2, 3] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor action | System Response | | 1 | Click on “Close” button. | System close dialog and cancel update process. | | 2 | Click on “Update” button. | Application will be updated. | | 3 | Click on “Cancel” button. | Confirmation dialog will be canceled and no change is made. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Click on “Update” button. | “New Field Name” is empty or longer than 100 words. Show error message: “Field Name is limited from 1 to 100 characters”. |   **Relationships:** N/A.  **Business Rules:**   * New information of field will be stored in storage. * System must ensure has not duplicated field name in available class. | | | |

**Table: Update Application specification**

## **Software System Attribute**

### **3.1. Usability**

#### **3.1.1. Graphic user interface**

All the test, labels and alerts of web application and mobile framework will be written by English.

#### **3.1.2. Usability**

The system usability is easy to use that will need less than 1 hour for training to use system.

#### **3.1.3. Installation**

User can follow installation and manual guide for installation. If there are any problems, user cans contacts developer for help.

### **3.2. Reliability**

* The data should be backed up everyday.

### **3.3. Availability**

* The service relates to communication so it can be available 24/7.
* Server should have back-up method to make sure that if it having problems, all necessary data can be protected and restore easily.

### **3.4. Security**

* Input data is validated if necessary before saving to database.
* Users is authenticated/authorized for all users when they logged in to the system.

### **3.5. Maintainability**

* System is separated into modules.

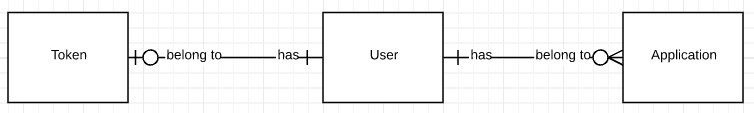
### **3.6. Portability**

* User, guest can use application on every OS supported web browsers.
* User can use mobile framework for develop any Android or iOS application.

### **3.7. Performance**

* Requests from web application are responded in less than 10 seconds at 8 Mbps bandwidth speed.

## **Conceptual Diagram**



**Figure: Conceptual Diagram**

|  |  |
| --- | --- |
| **Entity Data dictionary: describe all content of all entities** | |
| **Entity Name** | **Description** |
| User | Contain the user information |
| Token | Contain the token information |
| Application | Contain the application information |

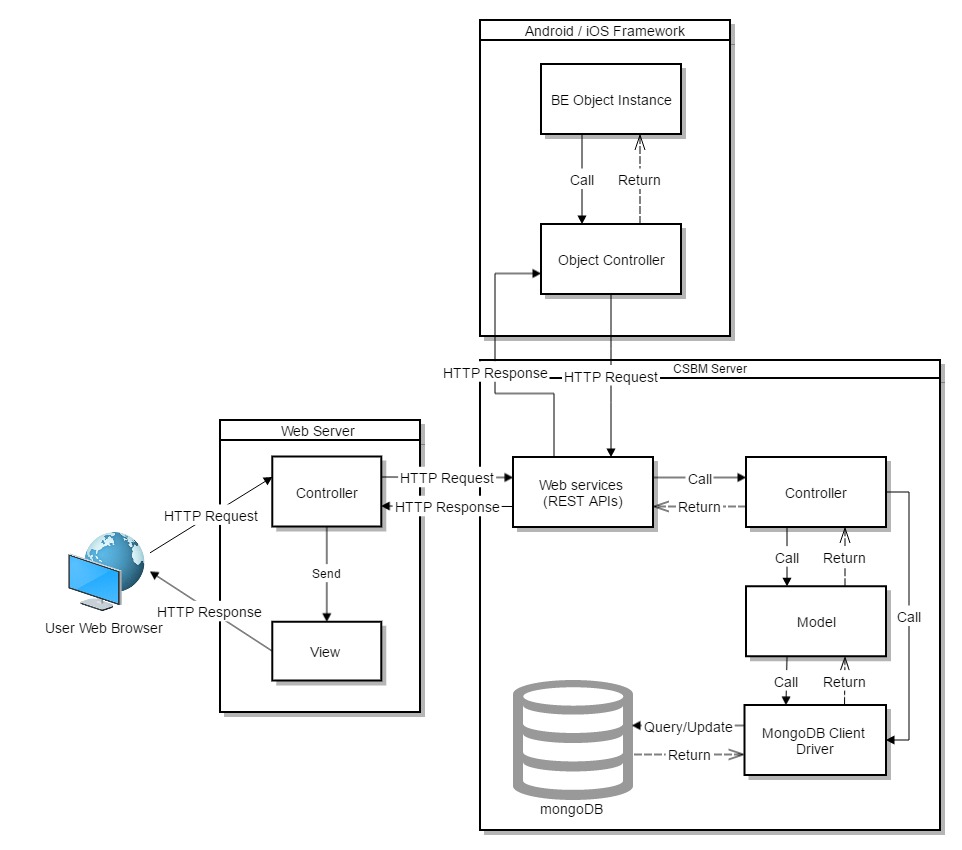
**Table: Conceptual diagram data dictionary**

# **D. Software Design Description**

## **Design Overview**

* This document describes the technical and user interface design of **CSBM System**. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
  + - Section 2: gives an overall description of the system architecture design.
    - Section 3: gives component diagrams that describe the connection and integration of the system.
    - Section 4: gives the detail design description, which includes class diagram, class explanation, and sequence diagram to details the application functions.
    - Section 5: describe screens design.
    - Section 6: describe a fully attributed ERD.
    - Section 7: describe algorithms.

## **System Architectural Design**



**Figure: System architecture design**

### **2.1 Web application architecture description**

In Web Application, the system is developed under AngularJS MVC architecture style. We choose this architecture for Web application because of following advantages:

* Web app contains a Web service with MVC architecture, we can separate business code with Controller and View, so we can use the business code in web service without repeat the code.
* We can organize the code better for maintainability, extensibility, reusability.
* In scope of 4-member team, MVC architecture make it easier to split the big project into small modules and make it easier to assign each module for members in our team.

The project follows MVC architecture with following components:

* **Controller** is the parts of the controller that acts like event handler to handles user interaction.
* **View** is the parts of the web server that acts like event handles the display of the data. The selection of View is under control of Controller.

### **2.2 CSBM application architecture description**

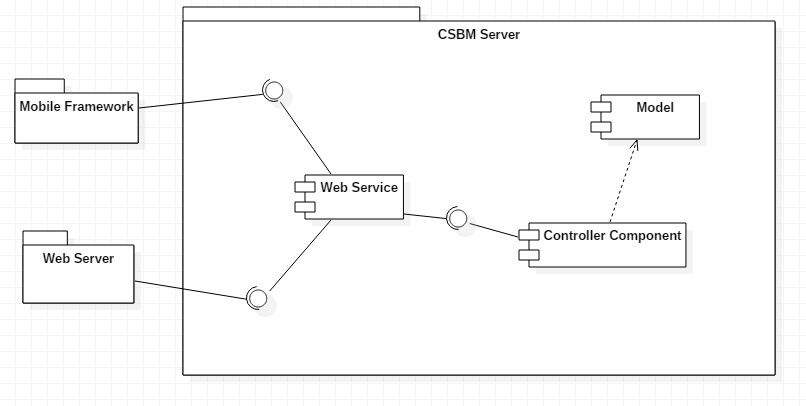
The application is developed under MVC architecture style.

This project follows MVC architecture with following component:

* **Web services** is the parts of the application that acts like event handler for web and mobile communication via REST method.
* **Controller** is the parts of the application that read data from a request and calls appropriate MongoDB Client Driver’s method then return to web services.
* **Model** is the parts of the application that acts like data transfer object between the system and database.
* **Web services** is the parts of the application that acts like event handler for web and mobile communication via REST method.
* **MongoDB Client Driver** is the parts of the application that provides some methods and classes to persistent store data into database.

### **2.3 Mobile framework architecture description**

## **Component Diagram**

****

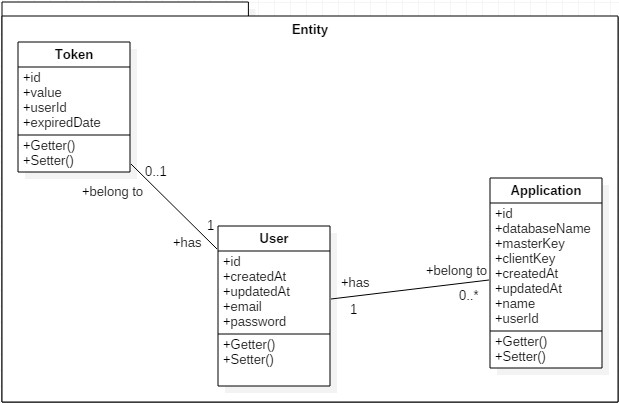
**Figure: Component Diagram**

|  |  |
| --- | --- |
| Component Dictionary: Describes components | |
| **Mobile Framework** | Mobile Framework package |
| **Web Server** | Web Server package: View, Controller |
| **Web Service** | Include all web API Controllers of the system |
| **Controller Component** | Component is used to handle request |
| **Model** |  |

**Table: Component Dictionary**

## **Detail Description**

### **4.1 Class Diagram**

****

**Figure: Class Diagram**

|  |  |  |
| --- | --- | --- |
| Class dictionary: describe Class | | |
| Class Name | **Mapping column with Conceptual diagram** | **Description** |
| **User** | User | Contain the user information |
| **Token** | Token | Contain the token information |
| **Application** | Application | Contain the application information |

**Table: Class Dictionary**

### **4.2 Class Diagram Explanation**

#### **4.2.1 User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| **id** | String | Private | Unique identifier of a user |
| **createdAt** | Date | Private | Date of create user |
| **updatedAt** | Date | Private | Date of update user |
| **email** | String | Private | User’s email |
| **password** | String | Private | User’s password |

**Table: Account Attributes**

#### **4.2.2 Token**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| **id** | String | Private | Unique identifier of a token |
| **value** | String | Private | Token’s value |
| **userId** | String | Private | User’s Id |
| **expiredDate** | Date | Private | Token’s expiredDate of license |

**Table: Token Attributes**

#### **4.2.3 Application**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| **id** | String | Private | Unique identifier of an application |
| **databaseName** | String | Private | Unique identifier of an application database |
| **masterKey** | String | Private | Application’s MasterKey |
| **clientKey** | String | Private | Application’s ClientKey |
| **createdAt** | Date | Private | Date of create application |
| **updatedAt** | Date | Private | Date of update application |
| **name** | String | Private | Application’s Name |
| **userId** | String | Private | User’s Id |

**Table: Application Attributes**

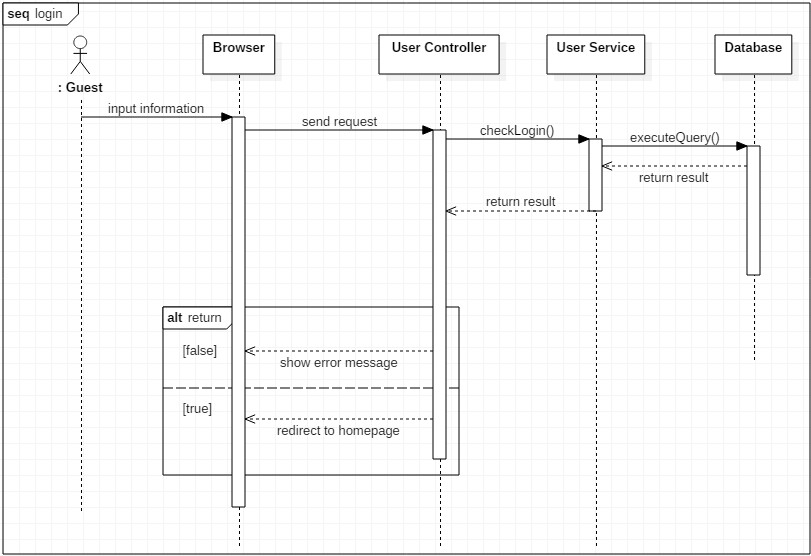
### **4.3 Interaction Diagram**

#### **4.3.1 Web Application**

##### **4.3.1.1 <Guest>**

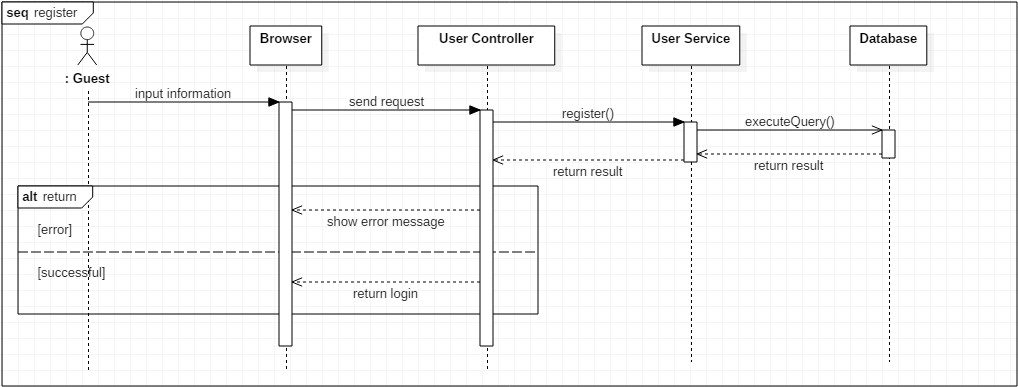
##### **4.3.1.1.1 Login**

Summary: This diagram show how a guest login system.

**Figure: Sequence Diagram - <Guest> Login**

##### **4.3.1.1.2 Register**

Summary: This diagram show process of register

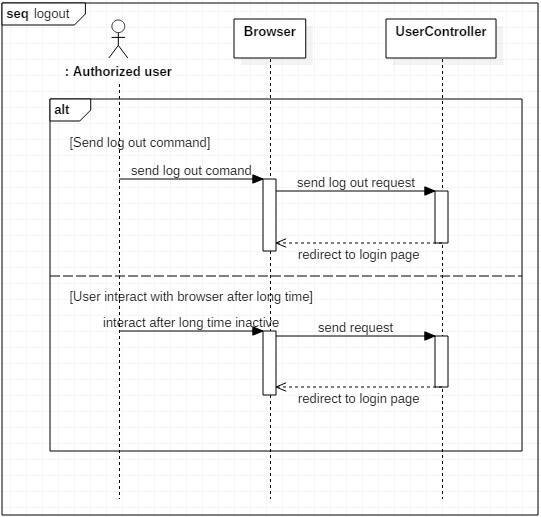


**Figure: Sequence Diagram - <Guest> Register**

##### **4.3.1.2 <Authorized User>**

##### **4.3.1.2.1 Logout**

Summary: This diagram show process of Logout

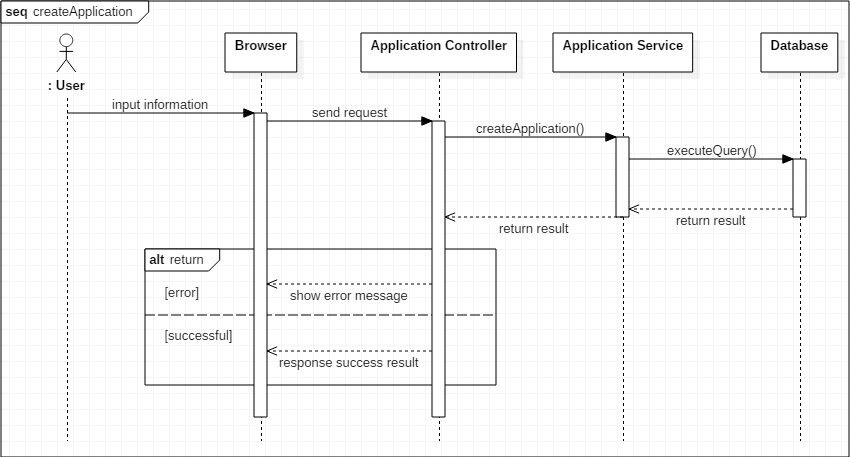
****

**Figure: Sequence Diagram - <Authorized User> Logout**

##### **4.3.1.3 <User>**

##### **4.3.1.3.1 Create New Application**

Summary: This diagram show process of Create New Application



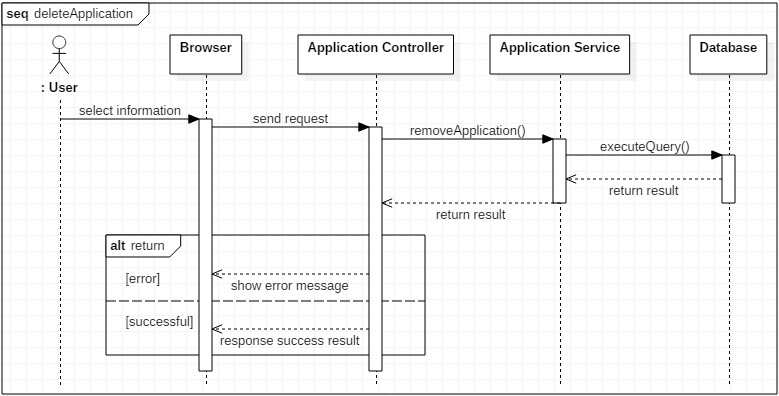
**Figure: Sequence Diagram - <User> Create New Application**

##### **4.3.1.3.2 Update Application**

Summary: This diagram show process of Update Application

##### **4.3.1.3.3 Delete Application**

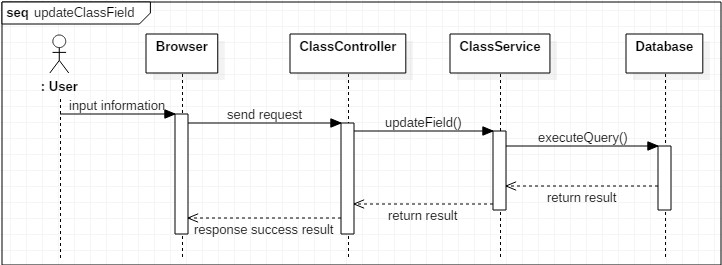
Summary: This diagram show process of Delete Application

****

**Figure: Sequence Diagram - <User> Delete Application**

##### **4.3.1.3.4 Update Class Field**

Summary: This diagram show process of Update Class Field

****

**Figure: Sequence Diagram - <User> Update Class Field**

##### **4.3.1.4 <Admin>**

##### **4.3.1.4.1 Change Application Status**

Summary: This diagram show process of Change Application Status

##### **4.3.1.4.2 Change User Status**

Summary: This diagram show process of Change User Status

## **Interface**

### **5.1 Web Service**

### **5.2 Mobile Framework APIs**

#### **iOS Framework APIs**

##### **5.2.1.1 CSBM**

|  |
| --- |
| **@interface** **CSBM** : **NSObject** |

The CSBM class contains static functions that handle global configuration for the CSBM framework.

###### **5.2.1.1.1 Connecting to CSBM**

* setApplicationId:clientKey

Sets the applicationId and clientKey of your application.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (void)setApplicationId:(nonnull NSString \*)applicationId  clientKey:(nonnull NSString \*)clientKey; |

**Parameters**

|  |  |
| --- | --- |
| *applicationID* | The application id of your CSBM application |
| *clientKey* | The client key of your CSBM application |

* initializeWithConfiguration

Sets the configuration to be used for the CSBM framework. Re-setting the configuration after having previously sent requests through the framework results in undefined behavior.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (void)**initializeWithConfiguration**:  (nonnull CSBMClientConfiguration \*)**configuration**; |

**Parameters**

|  |  |
| --- | --- |
| *applicationID* | The application id of your CSBM application |

* currentConfiguration

Gets the current configuration in use by the CSBM framework.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (nonnull CSBMClientConfiguration \*)  **currentConfiguration**; |

**Return Value**

|  |
| --- |
| The current configuration in use by the framework. Returns nil if the FRAMEWORK has not been initialized yet. |

* getApplicationId

The current application id that was used to configure CSBM framework.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (nonnull NSString \*)**getApplicationId**; |

* [getClientKey](https://parse.com/docs/ios/api/Classes/Parse.html" \l "/c:objc(cs)Parse(cm)getClientKey)

The current client key that was used to configure CSBM framework.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (nonnull NSString \*)**getClientKey**; |

###### **5.2.1.1.2 Enabling Local Datastore**

* enableLocalDatastore

Enable pinning in your application. This must be called before your application can use pinning. The recommended way is to call this method before +setApplicationId:clientKey:.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (void)enableLocalDatastore; |

* isLocalDatastoreEnabled

Flag that indicates whether Local Datastore is enabled.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (BOOL)isLocalDatastoreEnabled; |

**Return Value**

|  |
| --- |
| YES if Local Datastore is enabled, otherwise NO. |

###### **5.2.1.1.3 Enabling Extensions Data Sharing**

* enableDataSharingWithApplicationGroupIdentifier:

Enables data sharing with an application group identifier.

After enabling - Local Datastore, *BEUser.+currentUser*, *BEInstallation.+currentInstallation* and all eventually commands are going to be available to every application/extension in a group that have the same CSBM applicationId.

**Warning**

|  |
| --- |
| This method is required to be called before +setApplicationId:clientKey:. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (void)enableDataSharingWithApplicationGroupIdentifier:  (nonnull NSString \*)groupIdentifier; |

**Parameters**

|  |  |
| --- | --- |
| *groupIdentifier* | Application Group Identifier to share data with. |

* enableDataSharingWithApplicationGroupIdentifier:containingApplication:

Enables data sharing with an application group identifier.

Adter enabling – Local Datastore, BEUser.+currentUser, *BEInstallation.+currentInstallation* and all eventually commands are going to be available to every application/extension in a group that have the same CSBM applicationId.

**Warning**

|  |
| --- |
| This method is required to be called before +setApplicationId:clientKey:. This method can only be used by application extensions. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (void)enableDataSharingWithApplicationGroupIdentifier:  (nonnull NSString \*)groupIdentifier  containingApplication:  (nonnull NSString \*)bundleIdentifier; |

**Parameters**

|  |  |
| --- | --- |
| *groupIdentifier* | Application Group Identifier to share data with. |
| bundleIdentifier | Bundle identifier of the containing application. |

* applicationGroupIdentifierForDataSharing

Application Group Identifier for Data Sharing.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (nonnull NSString \*)  applicationGroupIdentifierForDataSharing; |

**Return Value**

NSString value if data sharing is enabled, otherwise nil.

* containingApplicationBundleIdentifierForDataSharing

Containing application bundle identifier for Data Sharing.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (nonnull NSString \*) containingApplicationBundleIdentifierForDataSharing; |

**Return Value**

NSString value if data sharing is enabled, otherwise nil.

##### **5.2.1.2 CSBMClientConfiguration**

|  |
| --- |
| **@interface** **CSBMClientConfiguration** : **NSObject** <NSCopying> |

The CSBMClientConfiguration represents the local configuration of the framework to connect to the server with.

These configurations can be stored, copied, and compared, but cannot be safely changed once the framework is initialized.

Use this object to construct a configuration for the SDK in your application, and pass it to CSBM.+initializeWithConfiguration:.

###### **5.2.1.2.1 Connecting to CSBM**

* applicationId

The CSBM application id to configure the framework with.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic, nullable)  NSString **\*applicationId**; |

* clientKey

The CSBM client key to configure the framework with.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic, nullable)  NSString **\*clientKey**; |

* server

The URL of the server that is being used by the framework.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic)  NSString **\*\_Nonnull server**; |

###### **5.2.1.2.2 Enabling Local Datastore**

localDatastoreEnabled

Whether or not to enable pinning in the framework.

The default value is NO.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, getter**=**isLocalDatastoreEnabled, assign, nonatomic)  BOOL **localDatastoreEnabled**; |

###### **5.2.1.2.3 Enabling Extensions Data Sharing**

* [applicationGroupIdentifier](https://parse.com/docs/ios/api/Classes/ParseClientConfiguration.html" \l "/c:objc(cs)ParseClientConfiguration(py)applicationGroupIdentifier)

When set, enables data sharing with an application group identifier.

After enabling - Local Datastore, PFUser.+currentUser, BEInstallation.+currentInstallation and all eventually commands are going to be available to every application/extension in a group that have the same Parse applicationId.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic, nullable)  NSString **\*applicationGroupIdentifier;** |

* containingApplicationBundleIdentifier

When set, controls the bundle identifier of the parent bundle to connect to.

**WARNING**

|  |
| --- |
| This property should only be set from inside an extension environment. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic, nullable)  NSString **\*containingApplicationBundleIdentifier;** |

###### **5.2.1.2.4 Other Properties**

* networkRetryAttempts

The maximum number of retry attempts to make upon a failed network request.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, assign, nonatomic)NSUInteger  **networkRetryAttempts;** |

###### **5.2.1.2.5 Creating a Configuration**

* configurationWithBlock:

Create a new FRAMEWORK configuration object. This will create a temporarily modifiable configuration, and pass it to a block to be initialized.

**Example usage:**

|  |
| --- |
| [CSBMClientConfiguration configurationWithBlock:^(id<CSBMMutableClientConfiguration> configuration) {  configuration.applicationId = ...;  configuration.clientKey = ...;  configuration.localDatastoreEnabled = ...;  }]; |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+ (nonnull instancetype)configurationWithBlock:**  **(nonnull void (^)(id<CSBMMutableClientConfiguration> \_Nonnull))**  **configurationBlock;** |

**Parameters**

|  |  |
| --- | --- |
| configurationBlock | A block used to modify the created configuration. |

**Return Value**

A newly created configuration.

##### **5.2.1.3 BEUser**

|  |
| --- |
| **@interface BEUser : BEObject** <BESubclassing> |

The BEUser class is a local representation of a user persisted to the CSBM Data. This class is a subclass of a [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html), and retains the same functionality of a [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html), but also extends it with various user specific methods, like authentication, signing up, and validation uniqueness.

Many APIs responsible for linking a BEUser with Facebook or Twitter have been deprecated in favor of dedicated utilities for each social network.

###### **5.2.1.3.1 Accessing the Current User**

* currentUser

Gets the currently logged in user from disk and returns an instance of it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  +(nullable instancetype)**currentUser;** |

**Return Value**

Returns a BEUser that is the currently logged in user. If there is none, returns nil.

[sessionToken](https://parse.com/docs/ios/api/Classes/PFUser.html" \l "/c:objc(cs)PFUser(py)sessionToken)

The session token for the BEUser.

This is set by the server upon successful authentication.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic, nullable) NSString  **\*sessionToken;** |

[isNew](https://parse.com/docs/ios/api/Classes/PFUser.html" \l "/c:objc(cs)PFUser(py)isNew)

Whether the [BEUser](https://parse.com/docs/ios/api/Classes/PFUser.html) was just created from a request.

This is only set after a Facebook login.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, assign, nonatomic)BOOL **isNew;** |

[authenticated](https://parse.com/docs/ios/api/Classes/PFUser.html" \l "/c:objc(cs)PFUser(py)authenticated)

Whether the user is an authenticated object for the device.

An authenticated BEUser is one that is obtained via a -signUp: or +logInWithUsername:password: method.

An authenticated object is required in order to save (with altered values) or delete it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, getter=isAuthenticated, assign,  nonatomic)BOOL **authenticated;** |

###### **5.2.1.3.2 Creating a New User**

* [user](https://parse.com/docs/ios/api/Classes/PFUser.html" \l "/c:objc(cs)PFUser(cm)user)

Creates a new BEUser object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  +(nonnull instancetype)**user;** |

**Return Value**

Returns a new BEUser object.

[username](https://parse.com/docs/ios/api/Classes/PFUser.html" \l "/c:objc(cs)PFUser(py)username)

The username for the BEUser.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readwrite, strong, nonatomic, nullable)  NSString **\*username;** |

[password](https://parse.com/docs/ios/api/Classes/PFUser.html" \l "/c:objc(cs)PFUser(py)password)

! The password for the BEUser.

This will not be filled in from the server with the password. It is only meant to be set.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readwrite, strong, nonatomic, nullable)  NSString **\*password;** |

email

The email for the BEUser.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readwrite, strong, nonatomic, nullable)  NSString **\*email;** |

* signUpInBackground

Signs up the user *asynchronously*.

This will also enforce that the username isn’t already taken.

**Warning**

|  |
| --- |
| Make sure that password and username are set before calling this method. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**signUpInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* signUpInBackgroundWithBlock:

Signs up the user *asynchronously*.

This will also enforce that the username isn’t already taken.

**Warning**

|  |
| --- |
| Make sure that password and username are set before calling this method. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**signUpInBackgroundWithBlock**:(nullable  BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.3.3 Logging In**

* logInWithUsernameInBackground:password:

Makes an *asynchronous* request to login a user with specified credentials.

Returns an instance of the successfully logged in [BEUser](https://parse.com/docs/ios/api/Classes/PFUser.html). This also caches the user locally so that calls to +currentUser will use the latest logged in user.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**logInWithUsernameInBackground:**(nonnull NSString \*)**username**  **password:**(nonnull NSString \*)**password;** |

**Parameters**

|  |  |
| --- | --- |
| username | The username of the user. |
| password | The password of the user. |

**Return Value**

The task, that encapsulates the work being done.

* logInWithUsernameInBackground:password:block:

Makes an *asynchronous* request to log in a user with specified credentials.

Returns an instance of the successfully logged in [BEUser](https://parse.com/docs/ios/api/Classes/PFUser.html). This also caches the user locally so that calls to [+currentUser](https://parse.com/docs/ios/api/Classes/PFUser.html#/c:objc(cs)PFUser(cm)currentUser) will use the latest logged in user.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**logInWithUsernameInBackground:**(nonnull NSString \*)**username**  **password:**(nonnull NSString \*)**password**  **block:**(nullable BEUserResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| username | The username of the user. |
| password | The password of the user. |
| block | The block to execute. It should have the following argument signature: ^(BEUser \*user, NSError \*error). |

###### **5.2.1.3.4 Becoming a User**

* becomeInBackground:

Makes an *asynchronous* request to become a user with the given session token.

Returns an instance of the successfully logged in [BEUser](https://parse.com/docs/ios/api/Classes/PFUser.html). This also caches the user locally so that calls to [+currentUser](https://parse.com/docs/ios/api/Classes/PFUser.html#/c:objc(cs)PFUser(cm)currentUser) will use the latest logged in user.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**becomeInBackground:**(nonnull NSString \*)**sessionToken;** |

**Parameters**

|  |  |
| --- | --- |
| sessionToken | The session token for the user. |

**Return Value**

The task, that encapsulates the work being done.

* becomeInBackground:block:

Makes an *asynchronous* request to become a user with the given session token.

Returns an instance of the successfully logged in [BEUser](https://parse.com/docs/ios/api/Classes/PFUser.html). This also caches the user locally so that calls to [+currentUser](https://parse.com/docs/ios/api/Classes/PFUser.html#/c:objc(cs)PFUser(cm)currentUser) will use the latest logged in user.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**becomeInBackground:**(nonnull NSString \*)**sessionToken**  **block:**(nullable BEUserResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| sessionToken | The session token for the user. |
| block | The block to execute. The block should have the following argument signature:^(BEUser \*user, NSError \*error). |

###### **5.2.1.3.5 Revocable Session**

* enableRevocableSessionInBackground

Enables revocable sessions and migrates the currentUser session token to use revocable session if needed.

This method is required if you want to use BESession APIs and your application’s ‘Require Revocable Session’ setting is turned off on http://csbm.com app settings. After returned BFTask completes - BESession class and APIs will be available for use.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**enableRevocableSessionInBackground;** |

**Return Value**

An instance of BFTask that is completed when revocable sessions are enabled and currentUser token is migrated.

* enableRevocableSessionInBackgroundWithBlock:

Enables revocable sessions and upgrades the currentUser session token to use revocable session if needed.

This method is required if you want to use BESession APIs and legacy sessions are enabled in your application settings on http://parse.com/. After returned BFTask completes - BESession class and APIs will be available for use.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**enableRevocableSessionInBackgroundWithBlock:**  (nullable BEUserSessionUpgradeResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | Block that will be called when revocable sessions are enabled and currentUser token is migrated. |

###### **5.1.2.3.6 Logging Out**

* logOutInBackground

*Asynchronously* logs out the currently logged in user.

This will also remove the session from disk, log out of linked services and all future calls to +currentUser will return nil. This is preferrable to using -logOut, unless your code is already running from a background thread.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**logOutInBackground;** |

**Return Value**

An instance of BFTask, that is resolved with nil result when logging out completes.

* logOutInBackgroundWithBlock:

*Asynchronously* logs out the currently logged in user.

This will also remove the session from disk, log out of linked services and all future calls to [+currentUser](https://parse.com/docs/ios/api/Classes/PFUser.html#/c:objc(cs)PFUser(cm)currentUser) will return nil. This is preferrable to using –logOut, unless your code is already running from a background thread.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**logOutInBackgroundWithBlock:**(nullable  BEUserLogoutResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | A block that will be called when logging out completes or fails. |

###### **5.1.2.3.7 Requesting a Password Reset**

* requestPasswordResetForEmailInBackground:

Send a password reset request *asynchronously* for a specified email and sets an error object. If a user account exists with that email, an email will be sent to that address with instructions on how to reset their password. - parameter: email Email of the account to send a reset password request. - returns: The task, that encapsulates the work being done.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**requestPasswordResetForEmailInBackground:**(nonnull  NSString \*)**email;** |

**Parameters**

|  |  |
| --- | --- |
| email | Email of the account to send a reset password request. |

**Return Value**

The task, that encapsulates the work being done.

* requestPasswordResetForEmailInBackground:block:

Send a password reset request *asynchronously* for a specified email.

If a user account exists with that email, an email will be sent to that address with instructions on how to reset their password.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+**(void)**requestPasswordResetForEmailInBackground:**(nonnull  NSString \*)**email**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| email | Email of the account to send a reset password request. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

##### **5.2.1.4 BEACL**

|  |
| --- |
| **@interface BEACL : NSObject** <NSCopying, NSCoding> |

The BEACL class is used to control which users can access or modify a particular object. Each [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) can have its own BEACL. Read and write permissions can be granted separately to specific users, to groups of users that belong to roles, or permissions to the public so that, for example, any user could read a particular object but only a particular set of users could write to that object.

###### **5.2.1.4.1 Creating an ACL**

* ACL

Creates an ACL with no permissions granted.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)**ACL;** |

**Return Value**

Returns a new BEACL.

* ACL With User

Creates an ACL where only the provided user has access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)**ACLWithUser**:  (nonnull BEUser \*)**user;** |

**Parameter**

|  |  |
| --- | --- |
| *user* | The user to assign access. |

###### **5.2.1.4.2 Controlling Public Access**

publicReadAccess

Controls whether the public is allowed to read this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@propertyv**(getter=getPublicReadAccess, assign,  readwrite, nonatomic) **BOOL publicReadAccess;** |

publicWriteAccess

Controls whether the public is allowed to write this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (getter=getPublicWriteAccess, assign,  readwrite, nonatomic) **BOOL publicWriteAccess;** |

###### **5.2.1.4.3 Controlling Access Per-User**

setReadAccess:forUserId:

Set whether the given user id is allowed to read this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**setReadAccess:**(BOOL)**allowed forUserId:**(nonnull  NSString \*)**userId;** |

**Parameters**

|  |  |
| --- | --- |
| *allowed* | Whether the given user can write this object. |
| *userId* | The BEObject.objectId of the user to assign access. |

* [getReadAccessForUserId:](https://parse.com/docs/ios/api/Classes/PFACL.html#/c:objc(cs)PFACL(im)getReadAccessForUserId:)

Gets whether the given user id is explicitly allowed to read this object. Even if this returns NO, the user may still be able to access it if [publicReadAccess](https://parse.com/docs/ios/api/Classes/PFACL.html#/c:objc(cs)PFACL(py)publicReadAccess) returns YES or if the user belongs to a role that has access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (BOOL)**getReadAccessForUserId:**(nonnull NSString \*)**userId;** |

**Parameters**

|  |  |
| --- | --- |
| *userId* | The BEObject.objectId of the user for which to retrive access. |

**Return Value**

YES if the user with this objectId has explicit read access, otherwise NO.

* setWriteAccess:forUserId:

Set whether the given user id is allowed to write this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**setWriteAccess:**(BOOL)**allowed forUserId:**(nonnull  NSString \*)**userId;** |

**Parameters**

|  |  |
| --- | --- |
| *userId* | The BEObject.objectId of the user to assign access. |

* getWriteAccessForUserId:

Gets whether the given user id is explicitly allowed to write this object. Even if this returns NO, the user may still be able to write it if [publicWriteAccess](https://parse.com/docs/ios/api/Classes/PFACL.html#/c:objc(cs)PFACL(py)publicWriteAccess) returns YES or if the user belongs to a role that has access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (BOOL)**getWriteAccessForUserId:**(nonnull NSString \*)**userId;** |

**Parameters**

|  |  |
| --- | --- |
| *userId* | The BEObject.objectId of the user to assign access. |

**Return Value**

YES if the user with this BEObject.objectId has explicit write access, otherwise NO.

* setReadAccess:forUser:

Set whether the given user is allowed to read this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  -(void)**setReadAccess**:(BOOL)**allowed forUser:**(nonnull BEUser\*)  **user;** |

**Parameters**

|  |  |
| --- | --- |
| *allowed* | Whether the given user can read this object. |
| *user* | The user to assign access. |

* getReadAccessForUser:

Gets whether the given user is explicitly allowed to read this object. Even if this returns NO, the user may still be able to access it if [publicReadAccess](https://parse.com/docs/ios/api/Classes/PFACL.html#/c:objc(cs)PFACL(py)publicReadAccess) returns YES or if the user belongs to a role that has access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (BOOL)**getReadAccessForUser:**(nonnull BEUser \*)**user;** |

**Parameters**

|  |  |
| --- | --- |
| *user* | The user for which to retrive access. |

**Return Value**

YES if the user has explicit read access, otherwise NO.

* setWriteAccess:forUser:

Set whether the given user is allowed to write this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (void)**setWriteAccess:**(BOOL)**allowed forUser:**(nonnull  BEUser \*)**user;** |

**Parameters**

|  |  |
| --- | --- |
| *allowed* | Whether the given user can write this object. |
| *user* | The user to assign access. |

* getWriteAccessForUser:

Gets whether the given user is explicitly allowed to write this object. Even if this returns NO, the user may still be able to write it if publicWriteAccess returns YES or if the user belongs to a role that has access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (BOOL)**getWriteAccessForUser:**(nonnull BEUser \*)**user;** |

**Parameters**

|  |  |
| --- | --- |
| *user* | The user to assign access. |

**Return Value**

YES if the user has explicit write access, otherwise NO.

###### **5.2.1.4.4 Controlling Access Per-Role**

* getReadAccessForRoleWithName:

Get whether users belonging to the role with the given name are allowed to read this object. Even if this returns NO, the role may still be able to read it if a parent role has read access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  -(BOOL)**getReadAccessForRoleWithName:**(nonnull NSString \*)  **name;** |

**Parameters**

|  |  |
| --- | --- |
| *name* | The name of the role. |

**Return Value**

YES if the role has read access, otherwise NO.

* setReadAccess:forRoleWithName:

Set whether users belonging to the role with the given name are allowed to read this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (void)**setReadAccess**:(BOOL)**allowed forRoleWithName:**(nonnull  NSString \*)**name**; |

**Parameters**

|  |  |
| --- | --- |
| *allowed* | Whether the given role can read this object. |
| *name* | The name of the role. |

* getWriteAccessForRoleWithName:

Get whether users belonging to the role with the given name are allowed to write this object. Even if this returns NO, the role may still be able to write it if a parent role has write access.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (BOOL)**getWriteAccessForRoleWithName:**(nonnull NSString \*)  **name;** |

**Parameters**

|  |  |
| --- | --- |
| *name* | The name of the role. |

**Return Value**

YES if the role has read access, otherwise NO.

* setWriteAccess:forRoleWithName:

Set whether users belonging to the role with the given name are allowed to write this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (void)**setWriteAccess**:(BOOL)**allowed forRoleWithName:**(nonnull  NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| *allowed* | Whether the given role can write this object. |
| *name* | The name of the role. |

* getReadAccessForRole:

Get whether users belonging to the given role are allowed to read this object. Even if this returns NO, the role may still be able to read it if a parent role has read access.

The role must already be saved on the server and it’s data must have been fetched in order to use this method.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (BOOL)**getReadAccessForRole**:(nonnull BERole \*)**role**; |

**Parameters**

|  |  |
| --- | --- |
| *name* | The name of the role. |

**Return Value**

YES if the role has read access, otherwise NO.

* setReadAccess:forRole:

Set whether users belonging to the given role are allowed to read this object.

The role must already be saved on the server and it’s data must have been fetched in order to use this method.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (void)**setReadAccess:**(BOOL)**allowed forRole:**(nonnull  BERole \*)**role;** |

**Parameters**

|  |  |
| --- | --- |
| *name* | The role to assign access. |

* getWriteAccessForRole:

Get whether users belonging to the given role are allowed to write this object. Even if this returns NO, the role may still be able to write it if a parent role has write access.

The role must already be saved on the server and it’s data must have been fetched in order to use this method.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (BOOL)**getWriteAccessForRole:**(nonnull BERole \*)**role**; |

**Parameters**

|  |  |
| --- | --- |
| *name* | The name of the role. |

**Return Value**

YES if the role has write access, otherwise NO.

* setWriteAccess:forRole:

Set whether users belonging to the given role are allowed to write this object.

The role must already be saved on the server and it’s data must have been fetched in order to use this method.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (void)**setWriteAccess:**(BOOL)**allowed forRole:**(nonnull  BERole \*)**role;** |

**Parameters**

|  |  |
| --- | --- |
| *allowed* | Whether the given role can write this object. |
| *name* | The role to assign access. |

###### **5.2.1.4.5 Setting Access Defaults**

* setDefaultACL:withAccessForCurrentUser:

Sets a default ACL that will be applied to all instances of PFObject when they are created.

* + If NO, the provided acl will be used without modification.
  + If acl is nil, this value is ignored.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (void)**setDefaultACL:**(nullable BEACL \*)**acl**  **withAccessForCurrentUser:**(BOOL)**currentUserAccess;** |

**Parameters**

|  |  |
| --- | --- |
| *acl* | The ACL to use as a template for all instance of BEObject created after this method has been called. This value will be copied and used as a template for the creation of new ACLs, so changes to the instance after this method has been called will not be reflected in new instance of BEObject. |
| *currentUserAccess* | * If YES, the BEACL that is applied to newly-created instance of BEObject will provide read and write access to the BEUser.+currentUser at the time of creation. * If NO, the provided acl will be used without modification. * If acl is nil, this value is ignored. |

##### **5.2.1.5 BEObject**

|  |
| --- |
| **@interface BEObject : NSObject** |

The BEObject class is a local representation of data persisted to the CSBM cloud. This is the main class that is used to interact with objects in your app.

###### **5.2.1.5.1 Creating a BEObject**

* initWithClassName:

Initializes a new empty BEObject instance with a class name.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  - (nonnull instancetype)**initWithClassName:**(nonnull  NSString \*)**newClassName;** |

**Parameters**

|  |  |
| --- | --- |
| newClassName | A class name can be any alphanumeric string that begins with a letter. It represents an object in your app, like a ‘User’ or a ‘Document’. |

**Return Value**

Returns the object that is instantiated with the given class name.

* objectWithClassName:

Creates a new BEObject with a class name.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  + (nonnull instancetype)**objectWithClassName:**(nonnull  NSString \*)**className;** |

**Parameters**

|  |  |
| --- | --- |
| className | A class name can be any alphanumeric string that begins with a letter. It represents an object in your app, like a ‘User’ or a ‘Document’. |

**Return Value**

Returns the object that is instantiated with the given class name.

* objectWithClassName:dictionary:

Creates a new [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) with a class name, initialized with data constructed from the specified set of objects and keys.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)  **objectWithClassName:**(nonnull NSString \*)**className**  **dictionary:**(nullable NSDictionary<NSString \*, id> \*)  **dictionary;** |

**Parameters**

|  |  |
| --- | --- |
| className | The object’s class. |
| dictionary | An NSDictionary of keys and objects to set on the new BEObject. |

**Return Value**

A BEObject with the given class name and set with the given data.

* objectWithoutDataWithClassName:objectId:

Creates a reference to an existing BEObject for use in creating associations between BEObjects.

Calling [dataAvailable](https://parse.com/docs/ios/api/Classes/PFObject.html#/c:objc(cs)PFObject(py)dataAvailable) on this object will return NO until -fetchIfNeeded has been called. No network request will be made.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)  **objectWithoutDataWithClassName:**(nonnull NSString \*)**className**  **objectId:**(nullable NSString \*)**objectId;** |

**Parameters**

|  |  |
| --- | --- |
| className | The object’s class. |
| objectId | The object id for the referenced object. |

**Return Value**

A [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) instance without data.

###### **5.2.1.5.2 Managing Object Properties**

CSBMClassName

The class name of the object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, strong, nonatomic) **NSString \*\_Nonnull**  **CSBMClassName;** |

objectId

The id of the object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readwrite, strong, nonatomic, nullable)  **NSString \*objectId;** |

updatedAt

When the object was last updated.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, strong, nonatomic, nullable) **NSDate**  **\*updatedAt;** |

createdAt

When the object was created.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, strong, nonatomic, nullable) **NSDate**  **\*createdAt;** |

ACL

The ACL for this object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readwrite, strong, nonatomic, nullable) **BEACL**  **\*ACL;** |

allKeys

Returns an array of the keys contained in this object.

This does not include createdAt, updatedAt, authData, or objectId. It does include things like username and ACL.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic) **NSArray<NSString \*>**  **\*\_Nonnull allKeys;** |

###### **5.2.1.5.3 Accessor**

* [objectForKey:](https://parse.com/docs/ios/api/Classes/PFObject.html" \l "/c:objc(cs)PFObject(im)objectForKey:)

Returns the value associated with a given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nullable id)**objectForKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key for which to return the corresponding value. |

* setObject:forKey:

Sets the object associated with a given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**setObject:**(nonnull id)**object forKey:**(nonnull  NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The object for key. A strong reference to the object is maintained by BEObject. Raises an NSInvalidArgumentException if object is nil. If you need to represent a nil value - use NSNull. |
| key | The key for object. Raises an NSInvalidArgumentException if key is nil. |

* removeObjectForKey:

Unsets a key on the object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**removeObjectForKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key. |

* objectForKeyedSubscript:

Returns the value associated with a given key.

This method enables usage of literal syntax on [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html). E.g. NSString \*value = object[@"key"];

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nullable id)**objectForKeyedSubscript:**(nonnull  NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key for which to return the corresponding value. |

* setObject:forKeyedSubscript:

Returns the value associated with a given key.

This method enables usage of literal syntax on BEObject. E.g. object[@"key"] = @"value";

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**setObject:**(nonnull id)**object forKeyedSubscript:**(nonnull  NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The object for key. A strong reference to the object is maintained by BEObject. Raises an NSInvalidArgumentException if object is nil. If you need to represent a nil value - use NSNull. |
| key | The key for object. Raises an NSInvalidArgumentException if key is nil. |

* relationForKey:

Returns the instance of BERelation class associated with the given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull BERelation \*)**relationForKey:**(nonnull  NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the relation is associated with. |

* revert

Clears any changes to this object made since the last call to save and sets it back to the server state.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**revert;** |

* revertObjectForKey:

Clears any changes to this object’s key that were done after last successful save and sets it back to the server state.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**revertObjectForKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to revert changes for. |

###### **5.2.1.5.4 Array Accessors**

* addObject:forKey:

Adds an object to the end of the array associated with a given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**addObject:**(nonnull id)**object forKey:**(nonnull  NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The object to add. |
| key | The key. |

* addObjectsFromArray:forKey:

Adds the objects contained in another array to the end of the array associated with a given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**addObjectsFromArray:**(nonnull NSArray \*)**objects**  **forKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The array of objects to add. |
| key | The key. |

* addUniqueObject:forKey:

Adds an object to the array associated with a given key, only if it is not already present in the array.

The position of the insert is not guaranteed.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**addUniqueObject:**(nonnull id)**object**  **forKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The object to add. |
| key | The key. |

* addUniqueObjectsFromArray:forKey:

Adds the objects contained in another array to the array associated with a given key, only adding elements which are not already present in the array.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**addUniqueObjectsFromArray:**(nonnull NSArray \*)**objects**  **forKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The array of objects to add. |
| key | The key. |

* removeObject:forKey:

Removes all occurrences of an object from the array associated with a given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**removeObject:**(nonnull id)**object forKey:**(nonnull  NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The object to remove. |
| key | The key. |

* removeObjectsInArray:forKey:

Removes all occurrences of the objects contained in another array from the array associated with a given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**removeObjectsInArray:**(nonnull NSArray \*)**objects**  **forKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| object | The array of objects to remove. |
| key | The key. |

###### **5.2.1.5.5 Increment**

* incrementKey:

Increments the given key by 1.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**incrementKey:**(nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key. |

* incrementKey:byAmount:

Increments the given key by a number.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**incrementKey:**(nonnull NSString \*)**key**  **byAmount:**(nonnull NSNumber \*)**amount;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key. |
| amount | The amount to increment. |

###### **5.2.1.5.6 Saving Objects**

saveInBackground

Saves the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) *asynchronously*.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**saveInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* saveInBackgroundWithBlock:

Saves the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) *asynchronously* and executes the given callback block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **- (void)saveInBackgroundWithBlock:(nullable**  **BEBooleanResultBlock)block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* saveEventually

Saves this object to the server at some unspecified time in the future, even if CSBM is currently inaccessible.

Use this when you may not have a solid network connection, and don’t need to know when the save completes. If there is some problem with the object such that it can’t be saved, it will be silently discarded.

Objects saved with this method will be stored locally in an on-disk cache until they can be delivered to CSBM. They will be sent immediately if possible. Otherwise, they will be sent the next time a network connection is available. Objects saved this way will persist even after the app is closed, in which case they will be sent the next time the app is opened. If more than 10MB of data is waiting to be sent, subsequent calls to -saveEventually will cause old saves to be silently discarded until the connection can be re-established, and the queued objects can be saved.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **- (id)saveEventually;** |

**Return Value**

The task that encapsulates the work being done.

* saveEventually:

Saves this object to the server at some unspecified time in the future, even if CSBM is currently inaccessible.

Use this when you may not have a solid network connection, and don’t need to know when the save completes. If there is some problem with the object such that it can’t be saved, it will be silently discarded. If the save completes successfully while the object is still in memory, then callback will be called.

Objects saved with this method will be stored locally in an on-disk cache until they can be delivered to CSBM. They will be sent immediately if possible. Otherwise, they will be sent the next time a network connection is available. Objects saved this way will persist even after the app is closed, in which case they will be sent the next time the app is opened. If more than 10MB of data is waiting to be sent, subsequent calls to -saveEventually: will cause old saves to be silently discarded until the connection can be re-established, and the queued objects can be saved.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**saveEventually:**(nullable BEBooleanResultBlock)**callback;** |

**Parameters**

|  |  |
| --- | --- |
| callback | The block to execute. It should have the following argument signature:  ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.5.7 Saving Many Objects**

* saveAllInBackground:

Saves a collection of objects all at once *asynchronously*.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**saveAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The array of objects to save. |

**Return Value**

The task that encapsulates the work being done.

* saveAllInBackground:block:

Saves a collection of objects all at once asynchronously and executes the block when done.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**saveAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The array of objects to save. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.5.8 Deleting Many Objects**

deleteAllInBackground:

Deletes a collection of objects all at once asynchronously. - parameter: objects The array of objects to delete. - returns: The task that encapsulates the work being done.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+ (id)deleteAllInBackground:(nullable NSArray<BEObject \*> \*)objects;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The array of objects to delete. |

**Return Value**

The task that encapsulates the work being done.

* deleteAllInBackground:block:

Deletes a collection of objects all at once *asynchronously* and executes the block when done.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)  **deleteAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The array of objects to delete. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.5.9 Getting an Object**

[dataAvailable](https://parse.com/docs/ios/api/Classes/PFObject.html" \l "/c:objc(cs)PFObject(py)dataAvailable)

Gets whether the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) has been fetched.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property (readonly, getter=isDataAvailable, assign, nonatomic)**  **BOOL dataAvailable;** |

**Return Value**

YES if the BEObject is new or has been fetched or refreshed, otherwise NO.

* fetchInBackground

Fetches the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) *asynchronously* and sets it as a result for the task.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**fetchInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* fetchInBackgroundWithBlock:

Fetches the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) *asynchronously* and executes the given callback block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **- (void)**  **fetchInBackgroundWithBlock:(nullable BEObjectResultBlock)block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BEObject \*object, NSError \*error). |

* fetchIfNeededInBackground

Fetches the BEObject data *asynchronously* if dataAvailable is NO, then sets it as a result for the task.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**fetchIfNeededInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* fetchIfNeededInBackgroundWithBlock:

Fetches the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) data *asynchronously* if [dataAvailable](https://parse.com/docs/ios/api/Classes/PFObject.html#/c:objc(cs)PFObject(py)dataAvailable) is NO, then calls the callback block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**fetchIfNeededInBackgroundWithBlock:**(nullable  BEObjectResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BEObject \*object, NSError \*error). |

###### **5.2.1.5.10 Getting Many Objects**

fetchAllInBackground:

Fetches all of the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) objects with the current data from the server *asynchronously*.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**fetchAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The list of objects to fetch. |

**Return Value**

The task that encapsulates the work being done.

* fetchAllInBackground:block:

Fetches all of the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) objects with the current data from the server *asynchronously* and calls the given block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**fetchAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **block:**(nullable BEArrayResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The list of objects to fetch. |
| block | The block to execute. It should have the following argument signature: ^(NSArray \*objects, NSError \*error). |

* fetchAllIfNeededInBackground:

Fetches all of the BEObject objects with the current data from the server *asynchronously*.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**fetchAllIfNeededInBackground:**(nullable  NSArray<BEObject \*> \*)**objects;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The list of objects to fetch. |

**Return Value**

The task that encapsulates the work being done.

* fetchAllIfNeededInBackground:block:

Fetches all of the BEObjects with the current data from the server *asynchronously* and calls the given block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)  **fetchAllIfNeededInBackground**:(nullable NSArray<BEObject \*> \*)**objects**  **block:**(nullable BEArrayResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The list of objects to fetch. |
| block | The block to execute. It should have the following argument signature: ^(NSArray \*objects, NSError \*error). |

###### **5.2.1.5.11 Fetching From Local Datastore**

* fetchFromLocalDatastoreInBackground

*Asynchronously* loads data from the local datastore into this object, if it has not been fetched from the server already.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **- (id)fetchFromLocalDatastoreInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* fetchFromLocalDatastoreInBackgroundWithBlock:

*Asynchronously* loads data from the local datastore into this object, if it has not been fetched from the server already.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**fetchFromLocalDatastoreInBackgroundWithBlock:**  (nullable BEObjectResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BEObject \*object, NSError \*error). |

###### **5.2.1.5.12 Deleting an Object**

deleteInBackground

Deletes the BEObject *asynchronously*.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**deleteInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* deleteInBackgroundWithBlock:

Deletes the [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) *asynchronously* and executes the given callback block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**deleteInBackgroundWithBlock:**(nullable  BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* deleteEventually

Deletes this object from the server at some unspecified time in the future, even if CSBM is currently inaccessible.

Use this when you may not have a solid network connection, and don’t need to know when the delete completes. If there is some problem with the object such that it can’t be deleted, the request will be silently discarded.

Delete instructions made with this method will be stored locally in an on-disk cache until they can be transmitted to CSBM. They will be sent immediately if possible. Otherwise, they will be sent the next time a network connection is available. Delete requests will persist even after the app is closed, in which case they will be sent the next time the app is opened. If more than 10MB of -saveEventually or -deleteEventually commands are waiting to be sent, subsequent calls to -saveEventually or -deleteEventually will cause old requests to be silently discarded until the connection can be re-established, and the queued requests can go through.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**deleteEventually;** |

**Return Value**

The task that encapsulates the work being done.

###### **5.2.1.5.13 Pinning**

* pinInBackground

*Asynchronously* stores the object and every object it points to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call -fetchFromLocalDatastore on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**pinInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* pinInBackgroundWithBlock:

*Asynchronously* stores the object and every object it points to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call -fetchFromLocalDatastore on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**pinInBackgroundWithBlock:**(nullable  BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* pinInBackgroundWithName:

*Asynchronously* stores the object and every object it points to in the local datastore, recursively.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call -fetchFromLocalDatastore on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**pinInBackgroundWithName:**(nonnull NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the pin. |

**Return Value**

The task that encapsulates the work being done.

* pinInBackgroundWithName:block:

*Asynchronously* stores the object and every object it points to in the local datastore, recursively.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call -fetchFromLocalDatastore on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**pinInBackgroundWithName:**(nonnull NSString \*)**name**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the pin. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.5.14 Pinning Many Objects**

pinAllInBackground:

*Asynchronously* stores the objects and every object they point to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call fetchFromLocalDatastore: on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**pinAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects to be pinned. |

**Return Value**

The task that encapsulates the work being done.

* pinAllInBackground:block:

*Asynchronously* stores the objects and every object they point to in the local datastore, recursively, using a default pin name: [BEObjectDefaultPin](https://parse.com/docs/ios/api/Constants.html#/c:@PFObjectDefaultPin).

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call fetchFromLocalDatastore: on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**pinAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects to be pinned. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* pinAllInBackground:withName:

*Asynchronously* stores the objects and every object they point to in the local datastore, recursively.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a BEQuery that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call fetchFromLocalDatastore: on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**pinAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **withName:**(nonnull NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects to be pinned. |
| name | The name of the pin. |

**Return Value**

The task that encapsulates the work being done.

* pinAllInBackground:withName:block:

*Asynchronously* stores the objects and every object they point to in the local datastore, recursively.

If those other objects have not been fetched from CSBM, they will not be stored. However, if they have changed data, all the changes will be retained. To get the objects back later, you can use a [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) that uses BEQuery.-fromLocalDatastore, or you can create an unfetched pointer with +objectWithoutDataWithClassName:objectId: and then call fetchFromLocalDatastore: on it.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**pinAllInBackground**:(nullable NSArray<BEObject \*> \*)**objects**  **withName**:(nonnull NSString \*)**name**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects to be pinned. |
| name | The name of the pin. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.5.15 Unpinning**

unpinInBackground

*Asynchronously* removes the object and every object it points to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**unpinInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* unpinInBackgroundWithBlock:

*Asynchronously* removes the object and every object it points to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**unpinInBackgroundWithBlock:**(nullable  BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* unpinInBackgroundWithName:

*Asynchronously* removes the object and every object it points to in the local datastore, recursively.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**unpinInBackgroundWithName:**(nonnull NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the pin. |

**Return Value**

The task that encapsulates the work being done.

* unpinInBackgroundWithName:block:

*Asynchronously* removes the object and every object it points to in the local datastore, recursively.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)  **unpinInBackgroundWithName:**(nonnull NSString \*)**name**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the pin. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

###### **5.2.1.5.16 Unpinning Many Objects**

unpinAllObjectsInBackground

*Asynchronously* removes all objects in the local datastore using a default pin name: [BEObjectDefaultPin](https://parse.com/docs/ios/api/Constants.html#/c:@PFObjectDefaultPin).

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**unpinAllObjectsInBackground;** |

**Return Value**

The task that encapsulates the work being done.

* unpinAllObjectsInBackgroundWithBlock:

Asynchronously removes all objects in the local datastore using a default pin name: [BEObjectDefaultPin](https://parse.com/docs/ios/api/Constants.html#/c:@PFObjectDefaultPin).

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**unpinAllObjectsInBackgroundWithBlock:**  (nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* unpinAllObjectsInBackgroundWithName:

*Asynchronously* removes all objects with the specified pin name.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**unpinAllObjectsInBackgroundWithName:**(nonnull NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the pin. |

**Return Value**

The task that encapsulates the work being done.

* unpinAllObjectsInBackgroundWithName:block:

*Asynchronously*removes all objects with the specified pin name*.*

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+ (void)unpinAllObjectsInBackgroundWithName:**  **(nonnull NSString \*)name block:(nullable BEBooleanResultBlock)block;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the pin. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* unpinAllInBackground:

*Asynchronously* removes the objects and every object they point to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+ (id)unpinAllInBackground:(nullable NSArray<BEObject \*> \*)objects;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects. |

**Return Value**

The task that encapsulates the work being done.

* unpinAllInBackground:block:

*Asynchronously* removes the objects and every object they point to in the local datastore, recursively, using a default pin name: BEObjectDefaultPin.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**unpinAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* unpinAllInBackground:withName:

*Asynchronously* removes the objects and every object they point to in the local datastore, recursively.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (id)**unpinAllInBackground**:(nullable NSArray<BEObject \*> \*)**objects**  **withName:**(nonnull NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects. |
| name | The name of the pin. |

**Return Value**

The task that encapsulates the work being done.

* unpinAllInBackground:withName:block:

*Asynchronously* removes the objects and every object they point to in the local datastore, recursively.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**unpinAllInBackground:**(nullable NSArray<BEObject \*> \*)**objects**  **withName:**(nonnull NSString \*)**name**  **block:**(nullable BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| objects | The objects. |
| name | The name of the pin. |
| block | The block to execute. It should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

##### **5.2.1.6 BEFile**

|  |
| --- |
| **@interface BEFile : NSObject** |

BEFile representes a file of binary data stored on the CSBM servers. This can be a image, video, or anything else that an application needs to reference in a non-relational way.

###### **5.2.1.6.1 Creating a BEFile**

* init

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**init;** |

* fileWithData:

Creates a file with given data. A name will be assigned to it by the server.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nullable instancetype)**fileWithData:**(nonnull NSData \*)**data;** |

**Parameters**

|  |  |
| --- | --- |
| data | The contents of the new BEFile. |

**Return Value**

A new BEFile.

* fileWithName:data:

Creates a file with given data and name.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nullable instancetype)**fileWithName:**(nullable NSString \*)**name**  **data:**(nonnull NSData \*)**data;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the new BEFile. The file name must begin with and alphanumeric character, and consist of alphanumeric characters, periods, spaces, underscores, or dashes. |
| data | The contents of the new BEFile. |

**Return Value**

A new [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) object.

* fileWithName:contentsAtPath:

Creates a file with the contents of another file.

**Warning**

|  |
| --- |
| This method raises an exception if the file at path is not accessible or if there is not enough disk space left. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nullable instancetype)**fileWithName:**(nullable NSString \*)**name**  **contentsAtPath:**(nonnull NSString \*)**path;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the new BEFile. The file name must begin with and alphanumeric character, and consist of alphanumeric characters, periods, spaces, underscores, or dashes. |
| path | The path to the file that will be uploaded to CSBM. |

**Return Value**

A new [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) instance.

* fileWithName:contentsAtPath:error:

Creates a file with the contents of another file.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nullable instancetype)**fileWithName:**(nullable NSString \*)**name**  **contentsAtPath:**(nonnull NSString \*)**path**  **error:**(NSError \*\_Nullable \*\_Nullable)**error;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the new BEFile. The file name must begin with and alphanumeric character, and consist of alphanumeric characters, periods, spaces, underscores, or dashes. |
| path | The path to the file that will be uploaded to CSBM. |
| error | On input, a pointer to an error object. If an error occurs, this pointer is set to an actual error object containing the error information. You may specify nil for this parameter if you do not want the error information. |

**Return Value**

A new [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) instance or nil if the error occured.

* fileWithName:data:contentType:

Creates a file with given data, name and content type.

**Warning**

|  |
| --- |
| This method raises an exception if the data supplied is not accessible or could not be saved. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nullable instancetype)**fileWithName:**(nullable NSString \*)**name**  **data:**(nonnull NSData \*)**data**  **contentType:**(nullable NSString \*)**contentType;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the new BEFile. The file name must begin with and alphanumeric character, and consist of alphanumeric characters, periods, spaces, underscores, or dashes. |
| data | The contents of the new BEFile. |
| contentType | Represents MIME type of the data. |

**Return Value**

A new [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) instance.

* fileWithName:data:contentType:error:

Creates a file with given data, name and content type.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nullable instancetype)**fileWithName:**(nullable NSString \*)**name**  **data:**(nonnull NSData \*)**data**  **contentType:**(nullable NSString \*)**contentType**  **error:**(NSError \*\_Nullable \*\_Nullable)**error;** |

**Parameters**

|  |  |
| --- | --- |
| name | The name of the new BEFile. The file name must begin with and alphanumeric character, and consist of alphanumeric characters, periods, spaces, underscores, or dashes. |
| data | The contents of the new BEFile. |
| contentType | Represents MIME type of the data. |

**Return Value**

A new BEFile instance or nil if the error occured.

* fileWithData:contentType:

Creates a file with given data and content type.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)**fileWithData:**(nonnull NSData \*)**data**  **contentType:**(nullable NSString \*)**contentType;** |

**Parameters**

|  |  |
| --- | --- |
| data | The contents of the new BEFile. |
| contentType | Represents MIME type of the data. |

**Return Value**

A new BEFile object.

###### **5.2.1.6.2 File Properties**

[name](https://parse.com/docs/ios/api/Classes/PFFile.html" \l "/c:objc(cs)PFFile(py)name)

The name of the file.

Before the file is saved, this is the filename given by the user. After the file is saved, that name gets prefixed with a unique identifier.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property (readonly, copy, nonatomic) NSString \*\_Nonnull name;** |

url

The url of the file.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, copy, nonatomic, nullable) **NSString \*url;** |

dirty

Whether the file has been uploaded for the first time.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, getter=isDirty, assign, nonatomic) **BOOL dirty;** |

###### **5.2.1.6.3 Storing Data with CSBM**

* saveInBackground

Saves the file *asynchronously*.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**saveInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* saveInBackgroundWithProgressBlock:

Saves the file *asynchronously*

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**saveInBackgroundWithProgressBlock**:(nullable  BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| progressBlock | The block should have the following argument signature: ^(int percentDone) |

**Return Value**

The task, that encapsulates the work being done.

* saveInBackgroundWithBlock:

Saves the file *asynchronously* and executes the given block.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**saveInBackgroundWithBlock**:(nullable  BEBooleanResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block should have the following argument signature: ^(BOOL succeeded, NSError \*error). |

* saveInBackgroundWithBlock:progressBlock:

Saves the file *asynchronously* and executes the given block.

This method will execute the progressBlock periodically with the percent progress. progressBlock will get called with 100 before resultBlock is called.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)  **saveInBackgroundWithBlock**:(nullable BEBooleanResultBlock)**block**  **progressBlock**:(nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block should have the following argument signature: ^(BOOL succeeded, NSError \*error). |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

###### **5.2.1.6.4 Getting Data from CSBM**

dataAvailable

Whether the data is available in memory or needs to be downloaded.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, getter=isDataAvailable, assign, nonatomic)  **BOOL dataAvailable;** |

* getDataInBackground

This method is like -getData but it fetches asynchronously to avoid blocking the current thread.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getDataInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* getDataInBackgroundWithProgressBlock:

This method is like -getData but it fetches asynchronously to avoid blocking the current thread.

This can help applications with many large files avoid memory warnings.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getDataInBackgroundWithProgressBlock:**  (nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

**Return Value**

The task, that encapsulates the work being done.

* getDataStreamInBackground

This method is like -getDataInBackground but avoids ever holding the entire [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) contents in memory at once.

This can help applications with many large files avoid memory warnings.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getDataStreamInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* getDataDownloadStreamInBackground

This method is like -getDataStreamInBackground, but yields a live-updating stream.

Instead of -getDataStream, which yields a stream that can be read from only after the request has completed, this method gives you a stream directly written to by the HTTP session. As this stream is not pre-buffered, it is strongly advised to use the NSStreamDelegate methods, in combination with a run loop, to consume the data in the stream, to do proper async file downloading.

**Note**

|  |
| --- |
| You MUST open this stream before reading from it.  Do NOT call waitUntilFinished on this task from the main thread. It may result in a deadlock. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getDataDownloadStreamInBackground;** |

**Return Value**

A task that produces a *live* stream that is being written to with the data from the server.

* getDataStreamInBackgroundWithProgressBlock:

This method is like [-getDataInBackground](https://parse.com/docs/ios/api/Classes/PFFile.html#/c:objc(cs)PFFile(im)getDataInBackground) but avoids ever holding the entire [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) contents in memory at once.

This can help applications with many large files avoid memory warnings. - parameter: progressBlock The block should have the following argument signature: ^(int percentDone)

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getDataStreamInBackgroundWithProgressBlock:**  (nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

**Return Value**

The task, that encapsulates the work being done.

* getDataDownloadStreamInBackgroundWithProgressBlock:

This method is like -getDataStreamInBackgroundWithProgressBlock:, but yields a live-updating stream.

Instead of -getDataStream, which yields a stream that can be read from only after the request has completed, this method gives you a stream directly written to by the HTTP session. As this stream is not pre-buffered, it is strongly advised to use the NSStreamDelegate methods, in combination with a run loop, to consume the data in the stream, to do proper async file downloading.

**Note**

|  |
| --- |
| You MUST open this stream before reading from it.  Do NOT call waitUntilFinished on this task from the main thread. It may result in a deadlock. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getDataDownloadStreamInBackgroundWithProgressBlock:**  (nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

**Return Value**

A task that produces a *live* stream that is being written to with the data from the server.

* getDataInBackgroundWithBlock:

*Asynchronously* gets the data from cache if available or fetches its contents from the network.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**getDataInBackgroundWithBlock:**(nullable  BEDataResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block should have the following argument signature: ^(NSData \*result, NSError \*error). |

* getDataStreamInBackgroundWithBlock:

This method is like -getDataInBackgroundWithBlock: but avoids ever holding the entire [BEFile](https://parse.com/docs/ios/api/Classes/PFFile.html) contents in memory at once.

This can help applications with many large files avoid memory warnings.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**getDataStreamInBackgroundWithBlock:**  (nullable BEDataStreamResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block should have the following argument signature: (NSInputStream \*result, NSError \*error). |

* getDataInBackgroundWithBlock:progressBlock:

*Asynchronously* gets the data from cache if available or fetches its contents from the network.

This method will execute the progressBlock periodically with the percent progress. progressBlock will get called with 100 before resultBlock is called.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)  **getDataInBackgroundWithBlock**:(nullable BEDataResultBlock)**resultBlock**  **progressBlock**:(nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| resultBlock | The block should have the following argument signature: ^(NSData \*result, NSError \*error). |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

* getDataStreamInBackgroundWithBlock:progressBlock:

This method is like -getDataInBackgroundWithBlock:progressBlock: but avoids ever holding the entire BEFile contents in memory at once.

This can help applications with many large files avoid memory warnings.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**getDataStreamInBackgroundWithBlock:**  (nullable BEDataStreamResultBlock)**resultBlock**  **progressBlock:**  (nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| resultBlock | The block should have the following argument signature:  ^(NSInputStream \*result, NSError \*error). |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

* getFilePathInBackground

*Asynchronously* gets the file path for file from cache if available or fetches its contents from the network.

**Note**

|  |
| --- |
| The file path may change between versions of SDK.  If you overwrite the contents of the file at returned path it will  persist those change until the file cache is cleared. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getFilePathInBackground;** |

**Return Value**

The task, with the result set to NSString representation of a file path.

* getFilePathInBackgroundWithProgressBlock:

*Asynchronously* gets the file path for file from cache if available or fetches its contents from the network.

**Note**

|  |
| --- |
| The file path may change between versions of SDK.  If you overwrite the contents of the file at returned path it will  persist those change until the file cache is cleared. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getFilePathInBackgroundWithProgressBlock:**  (nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

**Return Value**

The task, with the result set to NSString representation of a file path.

* getFilePathInBackgroundWithBlock:

*Asynchronously* gets the file path for file from cache if available or fetches its contents from the network.

**Note**

|  |
| --- |
| The file path may change between versions of SDK.  If you overwrite the contents of the file at returned path it will  persist those change until the file cache is cleared. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**getFilePathInBackgroundWithBlock:**  (nullable BEFilePathResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block should have the following argument signature: ^(NSString \*filePath, NSError \*error). |

* getFilePathInBackgroundWithBlock:progressBlock:

*Asynchronously* gets the file path for file from cache if available or fetches its contents from the network.

**Note**

|  |
| --- |
| The file path may change between versions of SDK.  If you overwrite the contents of the file at returned path it will  persist those change until the file cache is cleared. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**getFilePathInBackgroundWithBlock:**  (nullable BEFilePathResultBlock)**block**  **progressBlock:**  (nullable BEProgressBlock)**progressBlock;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block should have the following argument signature: ^(NSString \*filePath, NSError \*error). |
| progressBlock | The block should have the following argument signature: ^(int percentDone). |

###### **5.2.1.6.5 Interrupting a Transfer**

* cancel

Cancels the current request (upload or download of file).

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**cancel;** |

##### **5.2.1.7 BEQuery**

|  |
| --- |
| **@interface**  **BEQuery** < BEGenericObject : BEObject \* > **: NSObject**<NSCopying> |

The BEQuery class defines a query that is used to query for [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html)s.

###### **5.2.1.7.1 Creating a Query for a Class**

* initWithClassName:

Initializes the query with a class name.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **- (nonnull instancetype)initWithClassName:**  **(nonnull NSString \*)className;** |

**Parameters**

|  |  |
| --- | --- |
| className | The class name. |

* queryWithClassName:

Returns a [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) for a given class.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+ (nonnull instancetype)queryWithClassName:**  **(nonnull NSString \*)className;** |

**Parameters**

|  |  |
| --- | --- |
| className | The class to query on. |

**Return Value**

A [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) object.

* queryWithClassName:predicate:

Creates a BEQuery with the constraints given by predicate.

The following types of predicates are supported:

* Simple comparisons such as =, !=, <, >, <=, >=, and BETWEEN with a key and a constant.
* Containment predicates, such as x IN {1, 2, 3}.
* Key-existence predicates, such as x IN SELF.
* BEGINSWITH expressions.
* Compound predicates with AND, OR, and NOT.
* SubQueries with key IN %@, subquery.

The following types of predicates are NOT supported:

* Aggregate operations, such as ANY, SOME, ALL, or NONE.
* Regular expressions, such as LIKE, MATCHES, CONTAINS, or ENDSWITH.
* Predicates comparing one key to another.
* Complex predicates with many ORed clauses.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)**queryWithClassName:**  (nonnull NSString \*)**className**  **predicate:**  (nullable NSPredicate \*)**predicate;** |

**Parameters**

|  |  |
| --- | --- |
| className | The class to query on. |
| predicate | The predicate to create conditions from. |

CSBMClassName

The class name to query for.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readwrite, strong, nonatomic)  **NSString \*\_Nonnull CSBMClassName;** |

###### **5.2.1.7.2 Adding Basic Constraints**

* includeKey:

Make the query include BEObjects that have a reference stored at the provided key.

This has an effect similar to a join. You can use dot notation to specify which fields in the included object are also fetch.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**includeKey:**  (nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to load child BEObjects for. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* selectKeys:

Make the query restrict the fields of the returned [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html)s to include only the provided keys.

If this is called multiple times, then all of the keys specified in each of the calls will be included.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**selectKeys:**  (nonnull NSArray<NSString \*> \*)**keys;** |

**Parameters**

|  |  |
| --- | --- |
| key | The keys to include in the result. |

**Return Value**

The same instance of [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) as the receiver. This allows method chaining.

* whereKeyExists:

Add a constraint that requires a particular key exists.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKeyExists:**  (nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that should exist. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKeyDoesNotExist:

Add a constraint that requires a key not exist.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKeyDoesNotExist:**  (nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that should not exist. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:equalTo:

Add a constraint to the query that requires a particular key’s object to be equal to the provided object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **equalTo**:(nonnull id)**object;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| object | The object that must be equalled. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:lessThan:

Add a constraint to the query that requires a particular key’s object to be less than the provided object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **lessThan**:(nonnull id)**object;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| object | The object that provides an upper bound. |

**Return Value**

The same instance of [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) as the receiver. This allows method chaining.

* whereKey:lessThanOrEqualTo:

Add a constraint to the query that requires a particular key’s object to be less than or equal to the provided object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **lessThanOrEqualTo**:(nonnull id)**object;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| object | The object that must be equalled. |

**Return Value**

The same instance of [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) as the receiver. This allows method chaining.

* whereKey:greaterThan:

Add a constraint to the query that requires a particular key’s object to be greater than the provided object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **greaterThan**:(nonnull id)**object;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| object | The object that must be equalled. |

**Return Value**

The same instance of [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) as the receiver. This allows method chaining.

* whereKey:greaterThanOrEqualTo:

Add a constraint to the query that requires a particular key’s object to be greater than or equal to the provided object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **greaterThanOrEqualTo**:(nonnull id)**object;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| object | The object that must be equalled. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:notEqualTo:

Add a constraint to the query that requires a particular key’s object to be not equal to the provided object.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **notEqualTo**:(nonnull id)**object;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| object | The object that must not be equalled. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:containedIn:

Add a constraint to the query that requires a particular key’s object to be contained in the provided array.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **containedIn**:(nonnull NSArray \*)**array;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| array | The possible values for the key’s object. |

**Return Value**

The same instance of [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) as the receiver. This allows method chaining.

* whereKey:notContainedIn:

Add a constraint to the query that requires a particular key’s object not be contained in the provided array.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **notContainedIn**:(nonnull NSArray \*)**array;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| array | The list of values the key’s object should not be. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:containsAllObjectsInArray:

Add a constraint to the query that requires a particular key’s array contains every element of the provided array.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **containsAllObjectsInArray**:(nonnull NSArray \*)**array;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| array | The array of values to search for. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

###### **5.2.1.7.3 Adding Location Constraints**

* whereKey:nearGeoPoint:

Add a constraint to the query that requires a particular key’s coordinates (specified via [BEGeoPoint](https://parse.com/docs/ios/api/Classes/PFGeoPoint.html)) be near a reference point.

Distance is calculated based on angular distance on a sphere. Results will be sorted by distance from reference point.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey:(nonnull NSString \*)key**  **nearGeoPoint**:(nonnull BEGeoPoint \*)**geopoint;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| geopoint | The reference point represented as a BEGeoPoint. |

**Return Value**

The same instance of [BEQuery](https://parse.com/docs/ios/api/Classes/PFQuery.html) as the receiver. This allows method chaining.

* whereKey:nearGeoPoint:withinMiles:

Add a constraint to the query that requires a particular key’s coordinates (specified via BEGeoPoint) be near a reference point and within the maximum distance specified (in miles).

Distance is calculated based on a spherical coordinate system. Results will be sorted by distance (nearest to farthest) from the reference point.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **nearGeoPoint**:(nonnull BEGeoPoint \*)**geopoint**  **withinMiles**:(double)**maxDistance;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| geopoint | The reference point represented as a BEGeoPoint. |
| maxDistance | Maximum distance in miles. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:nearGeoPoint:withinKilometers:

Add a constraint to the query that requires a particular key’s coordinates (specified via BEGeoPoint) be near a reference point and within the maximum distance specified (in kilometers).

Distance is calculated based on a spherical coordinate system. Results will be sorted by distance (nearest to farthest) from the reference point.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **nearGeoPoint**:(nonnull BEGeoPoint \*)**geopoint**  **withinKilometers**:(double)**maxDistance;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| geopoint | The reference point represented as a BEGeoPoint. |
| maxDistance | Maximum distance in kilometers. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:nearGeoPoint:withinRadians:

Add a constraint to the query that requires a particular key’s coordinates (specified via BEGeoPoint) be near a reference point and within the maximum distance specified (in radians). Distance is calculated based on angular distance on a sphere. Results will be sorted by distance (nearest to farthest) from the reference point.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **nearGeoPoint**:(nonnull BEGeoPoint \*)**geopoint**  **withinRadians**:(double)**maxDistance;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| geopoint | The reference point as a BEGeoPoint. |
| maxDistance | Maximum distance in radians. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:withinGeoBoxFromSouthwest:toNortheast:

Add a constraint to the query that requires a particular key’s coordinates (specified via BEGeoPoint) be contained within a given rectangular geographic bounding box.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **withinGeoBoxFromSouthwest**:(nonnull BEGeoPoint \*)**southwest**  **toNortheast**:(nonnull BEGeoPoint \*)**northeast;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to be constrained. |
| southwest | The lower-left inclusive corner of the box. |
| northeast | The upper-right inclusive corner of the box. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

###### **5.2.1.7.4 Adding String Constraints**

* whereKey:matchesRegex:

Add a regular expression constraint for finding string values that match the provided regular expression.

**Warning**

|  |
| --- |
| This may be slow for large datasets. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **matchesRegex**:(nonnull NSString \*)**regex;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the string to match is stored in. |
| regex | The regular expression pattern to match. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:matchesRegex:modifiers:

Add a regular expression constraint for finding string values that match the provided regular expression.

**Warning**

|  |
| --- |
| This may be slow for large datasets. |

i - Case insensitive search

m - Search across multiple lines of input

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **matchesRegex**:(nonnull NSString \*)**regex**  **modifiers**:(nullable NSString \*)**modifiers;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the string to match is stored in. |
| regex | The regular expression pattern to match. |
| modifiers | Any of the following supported PCRE modifiers: - i - Case insensitive search - m - Search across multiple lines of input |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:containsString:

Add a constraint for finding string values that contain a provided substring.

**Warning**

|  |
| --- |
| This will be slow for large datasets. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **containsString**:(nullable NSString \*)**substring;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the string to match is stored in. |
| substring | The substring that the value must contain. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:hasPrefix:

Add a constraint for finding string values that start with a provided prefix.

This will use smart indexing, so it will be fast for large datasets.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **hasPrefix**:(nullable NSString \*)**prefix;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the string to match is stored in. |
| prefix | The substring that the value must start with. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:hasSuffix:

Add a constraint for finding string values that end with a provided suffix.

**Warning**

|  |
| --- |
| This will be slow for large datasets. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **hasSuffix**:(nullable NSString \*)**suffix;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the string to match is stored in. |
| suffix | The substring that the value must end with. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

###### **5.2.1.7.5 Adding Subqueries**

* orQueryWithSubqueries:

Returns a BEQuery that is the or of the passed in queries.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (nonnull instancetype)**orQueryWithSubqueries:**  (nonnull NSArray<BEQuery \*> \*)**queries;** |

**Parameters**

|  |  |
| --- | --- |
| queries | The list of queries to or together. |

**Return Value**

An instance of BEQuery that is the or of the passed in queries.

* whereKey:matchesKey:inQuery:

Adds a constraint that requires that a key’s value matches a value in another key in objects returned by a sub query.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **matchesKey**:(nonnull NSString \*)**otherKey**  **inQuery**:(nonnull BEQuery \*)**query;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the value is stored. |
| otherKey | The key in objects in the returned by the sub query whose value should match. |
| query | The query to run. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:doesNotMatchKey:inQuery:

Adds a constraint that requires that a key’s value NOT match a value in another key in objects returned by a sub query.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **doesNotMatchKey**:(nonnull NSString \*)**otherKey**  **inQuery**:(nonnull BEQuery \*)**query;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the value is stored. |
| otherKey | The key in objects in the returned by the sub query whose value should match. |
| query | The query to run. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:matchesQuery:

Add a constraint that requires that a key’s value matches a BEQuery constraint.

**Warning**

|  |
| --- |
| This only works where the key’s values are BEObjects or arrays of BEObjects. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **matchesQuery**:(nonnull BEQuery \*)**query;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the value is stored in |
| query | The query the value should match |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* whereKey:doesNotMatchQuery:

Add a constraint that requires that a key’s value to not match a BEQuery constraint.

**Warning**

|  |
| --- |
| This only works where the key’s values are BEObjects or arrays of BEObjects. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**whereKey**:(nonnull NSString \*)**key**  **doesNotMatchQuery**:(nonnull BEQuery \*)**query;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key that the value is stored in |
| query | The query the value should not match |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

###### **5.2.1.7.6 Sorting**

* orderByAscending:

Sort the results in *ascending* order with the given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**orderByAscending:**  (nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to order by. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* addAscendingOrder:

Additionally sort in *ascending* order by the given key.

The previous keys provided will precedence over this key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**addAscendingOrder:**  (nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to order by. |

* orderByDescending:

Sort the results in *descending* order with the given key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**orderByDescending:**  (nonnull NSString \*)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to order by. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* addDescendingOrder:

Additionally sort in *descending* order by the given key.

The previous keys provided will precedence over this key.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**addDescendingOrder:**  (nonnull NSString **\***)**key;** |

**Parameters**

|  |  |
| --- | --- |
| key | The key to order by. |

* orderBySortDescriptor:

Sort the results using a given sort descriptor.

**Warning**

|  |
| --- |
| If a sortDescriptor has custom selector or comparator - they aren’t going to be used. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**orderBySortDescriptor:**  (nonnull NSSortDescriptor \*)**sortDescriptor;** |

**Parameters**

|  |  |
| --- | --- |
| sortDescriptor | The NSSortDescriptor to use to sort the results of the query. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* orderBySortDescriptors:

Sort the results using a given array of sort descriptors.

**Warning**

|  |
| --- |
| If a sortDescriptor has custom selector or comparator - they aren’t going to be used. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**orderBySortDescriptors:**  (nullable NSArray<NSSortDescriptor \*> \*)**sortDescriptors;** |

**Parameters**

|  |  |
| --- | --- |
| sortDescriptors | An array of NSSortDescriptor objects to use to sort the results of the query. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

###### **5.2.1.7.7 Getting Objects by ID**

* getObjectInBackgroundWithId:

Gets a BEObject asynchronously and calls the given block with the result.

**Warning**

|  |
| --- |
| This method mutates the query. It will reset limit to 1, skip to 0 and remove all conditions, leaving onlyobjectId. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getObjectInBackgroundWithId:**  (nonnull NSString \*)**objectId;** |

**Parameters**

|  |  |
| --- | --- |
| objectId | The id of the object that is being requested. |

**Return Value**

The task, that encapsulates the work being done.

* getObjectInBackgroundWithId:block:

Gets a BEObject asynchronously and calls the given block with the result.

**Warning**

|  |
| --- |
| This method mutates the query. It will reset limit to 1, skip to 0 and remove all conditions, leaving onlyobjectId. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)  **getObjectInBackgroundWithId**:(nonnull NSString \*)**objectId**  **block:**  (nullable void (^)(BEGenericObject \_Nullable,  NSError \*\_Nullable))**block;** |

**Parameters**

|  |  |
| --- | --- |
| objectId | The id of the object that is being requested. |
| block | The block to execute. The block should have the following argument signature:  ^(NSArray \*object, NSError \*error) |

###### **5.2.1.7.8 Getting all Matches for a Query**

* findObjectsInBackground

Finds objects *asynchronously* and sets the NSArray of [BEObject](https://parse.com/docs/ios/api/Classes/PFObject.html) objects as a result of the task.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**findObjectsInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* findObjectsInBackgroundWithBlock:

Finds objects *asynchronously* and calls the given block with the results.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**findObjectsInBackgroundWithBlock:**  (nullable BEQueryArrayResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature:  ^(NSArray \*objects, NSError \*error) |

###### **5.2.1.7.9 Getting the First Match in Query**

* getFirstObjectInBackground

Gets an object *asynchronously* and sets it as a result of the task.

**Warning**

|  |
| --- |
| This method mutates the query. It will reset the limit to 1. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**getFirstObjectInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* getFirstObjectInBackgroundWithBlock:

Gets an object *asynchronously* and calls the given block with the result.

**Warning**

|  |
| --- |
| This method mutates the query. It will reset the limit to 1. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**getFirstObjectInBackgroundWithBlock:**  (nullable void (^)(BEGenericObject \_Nullable,  NSError \*\_Nullable))**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(BEObject \*object, NSError \*error). result will be nil if error is set OR no object was found matching the query. error will be nil if result is set OR if the query succeeded, but found no results. |

###### **5.2.1.7.10 Counting the Mathches in a Query**

* countObjectsInBackground

Counts objects *asynchronously* and sets NSNumber with count as a result of the task.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (id)**countObjectsInBackground;** |

**Return Value**

The task, that encapsulates the work being done.

* countObjectsInBackgroundWithBlock:

Counts objects *asynchronously* and calls the given block with the counts.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**countObjectsInBackgroundWithBlock:**  (nullable BEIntegerResultBlock)**block;** |

**Parameters**

|  |  |
| --- | --- |
| block | The block to execute. It should have the following argument signature: ^(int count, NSError \*error) |

###### **5.2.1.7.11 Cancelling a Query**

* cancel

Cancels the current network request (if any). Ensures that callbacks won’t be called.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**cancel;** |

###### **5.2.1.7.12 Paginating Results**

limit

A limit on the number of objects to return. The default limit is 100, with a maximum of 1000 results being returned at a time.

**Warning**

|  |
| --- |
| If you are calling findObjects with limit = 1, you may find it easier to use getFirst instead. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property (assign, readwrite, nonatomic) NSInteger limit;** |

skip

The number of objects to skip before returning any.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (assign, readwrite, nonatomic) **NSInteger skip;** |

###### **5.2.1.7.13 Controlling Caching Behavior**

cachePolicy

The cache policy to use for requests.

Not allowed when Pinning is enabled.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (assign, readwrite, nonatomic)  **BECachePolicy cachePolicy;** |

maxCacheAge

The age after which a cached value will be ignored

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (assign, readwrite, nonatomic)  **NSTimeInterval maxCacheAge;** |

hasCachedResult

Returns whether there is a cached result for this query.

@result YES if there is a cached result for this query, otherwise NO.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (readonly, assign, nonatomic)  **BOOL hasCachedResult;** |

* clearCachedResult

Clears the cached result for this query. If there is no cached result, this is a noop.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (void)**clearCachedResult;** |

* clearAllCachedResults

Clears the cached results for all queries.

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **+** (void)**clearAllCachedResults;** |

###### **5.2.1.7.14 Query Source**

* fromLocalDatastore

Change the source of this query to all pinned objects.

**Warning**

|  |
| --- |
| Requires Local Datastore to be enabled. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**fromLocalDatastore;** |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* fromPin

Change the source of this query to the default group of pinned objects.

**Warning**

|  |
| --- |
| Requires Local Datastore to be enabled. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**fromPin;** |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* fromPinWithName:

Change the source of this query to a specific group of pinned objects.

**Warning**

|  |
| --- |
| Requires Local Datastore to be enabled. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**fromPinWithName:**  (nullable NSString \*)**name;** |

**Parameters**

|  |  |
| --- | --- |
| name | The pinned group. |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

* ignoreACLs

Ignore ACLs when querying from the Local Datastore.

This is particularly useful when querying for objects with Role based ACLs set on them.

**Warning**

|  |
| --- |
| Requires Local Datastore to be enabled. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **-** (nonnull instancetype)**ignoreACLs;** |

**Return Value**

The same instance of BEQuery as the receiver. This allows method chaining.

###### **[5.2.1.7.15 Advanced Settings](https://parse.com/docs/ios/api/Classes/PFQuery.html" \l "/Advanced%20Settings)**

[trace](https://parse.com/docs/ios/api/Classes/PFQuery.html" \l "/c:objc(cs)PFQuery(py)trace)

Whether or not performance tracing should be done on the query.

**Warning**

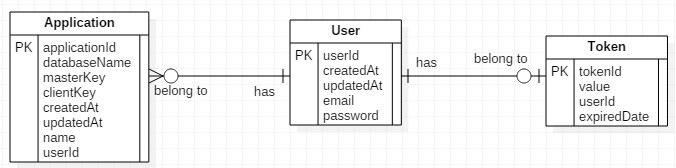
|  |
| --- |
| This should not be set to YES in most cases. |

**Declaration**

|  |
| --- |
| OBJECTIVE-C  **@property** (assign, readwrite, nonatomic) **BOOL trace;** |

## **Database Design**

### **6.1 Entity Relationship Diagram**

****

**Figure: Entity relationship diagram**

### **6.2 Data Dictionary**

|  |  |  |
| --- | --- | --- |
| **Entity Data Dictionary: describe content of all entities** | | |
| **Entity name** | **Description** | **Mapping column with Conceptual diagram** |
| **User** | Contain the user information | user |
| **Application** | Contain the application information | application |
| **Token** | Contain the token information | token |

**Table: Entity dictionary**

**< Mô tả các thành phần bên trong thực thể >**

**……………………..**

## **Algorithms**