Firebase

Firebase ist eine einfach zu bedienende Serveralternative, die Datenbank, Fileserver, Authentifizierung, Website Hosting und mehr über ein Webinterface vereint. Im Vergleich zu regulären Servern macht es Firebase einfacher für den Entwickler, sodass dieser mehr Zeit in sein Produkt selbst investieren kann.

Firebase bietet fortgeschrittene Sicherheitsmechanismen, Synchronisierung  
mehrerer Serverstandorte, Erreichbarkeit immer und überall und Offline Funktionalität.  
Alle Produkte sind mit verschiedensten Anmeldeoptionen verknüpfbar, wie Google Sign-In,  
Facebook Sign-In, Email/Passwort, Google Play, usw..

Datenbank

Firebase bietet hier zwei Möglichkeiten zur Speicherung von Daten, wobei beide auf NoSql-Datenbanken zurückgreifen. Daten können über das Webinterface angesehen, geändert und auch hinzugefügt werden.

Cloud Firestore

Diese neue Datenbanklösung ist zwar momentan noch in der Beta, ist jedoch von Beginn an für Skalierbarkeit optimiert. Wie für NoSql-Datenbanken üblich, wird enormer Fokus auf Lesegeschwindigkeiten und weniger auf Schreibgeschwindigkeiten gelegt. Verstärkt wird dieser Effekt dadurch, dass Indexe auf alle Felder gelegt werden um die gleichen Datenzugriffszeiten für 100 als auch für 100 Millionen Daten zu garantieren. Cloud Firestore bietet momentan SDK´s für Android und Ios Apps, Web, Node, Java, Python und Go an. Der Service verfügt unter Android, Ios und Web über die Möglichkeit, Daten lokal zu zwischenspeichern, wenn die Verbindung einmal verloren geht und diese dann mit dem Server zu synchronisieren, sobald das Problem behoben worden ist. Cloud Firestore ist eine dokumentenbasierte Lösung, was bedeutet das Objekte als Dokumente in einer Collection gespeichert werden. Diese Art der Speicherung wird oft als semistrukturierte Datenspeicherung bezeichnet. Dokumente haben verschiedene Felder, wobei Felder wieder Collections mit Dokumenten enthalten können. Diese sogenannten Subcollections ermöglichen in der Theorie zwar ein 1:1 Abbild der Datenstruktur aus Objektorientierten Sprachen, da diese jedoch nur einzeln gequeried werden können und nicht inkludiert werden können, verursacht diese Methode nur mehr Komplikationen. Eine Realisierung mittels Arrays von Referenzen, wäre hier die einfachere Variante, da Arrays inkludiert werden und, sofern das Programm das einlesen dieser händelt, einfach geparst werden können. Dieser Aufbau ähnelt sehr einer Relationalen Datenbankstruktur.

Realtime Database

Realtime Database bietet viele der selben Vorteile, wie Cloud Firestore. Nur der Fokus auf Skalierbarkeit ist nicht so stark ausgeprägt, jedoch ist das System stabiler, da es schon länger auf dem Markt verfügbar ist. SDK´s gibt es für Android, Ios, Web, C++ und Unity. Neben der besseren Verfügbarkeit der SDK für Unity war ein Mitentscheidungsgrund die Stabilität, die die Realtime Database bietet. Die Synchronisierung Offline geänderter Daten ist für Android und Ios verfügbar, nicht jedoch für Webanwendungen. Wir benutzen jedoch keine der drei Plattformen, und selbst wenn können unsere Nutzer keine Daten ändern oder einfügen, weshalb dieses Feature keine Bedeutung für uns hatte. Die Skalierbarkeit ist momentan noch kein Bedenken und der Umstieg auf Cloud Firestore ist auch während der Entwicklung noch leicht genug. Die Realtime Database benutzt die Key-Value Notation für die Datenspeicherung. Diese Schreibweise ähnelt JSON sehr, wodurch das einlesen und auslesen von Daten deutlich erleichtert wird.

Wir entschieden uns für die Realtime Database aus mehreren Gründen. Die Verfügbarkeit der SDK für Unity ist ein KO-Kriterium, da diese Technologie nicht änderbar ist und die Verwendung der Rest-Schnittstelle von Cloud Firestore außer Frage stand. Zusätzliche Features wie bessere Skalierbarkeit, bessere Offline Funktionalitäten oder strukturiertere Daten bringen wenig bis keinen Vorteil, da unsere Datenbank nur die Spiele speichert, die zur Verfügung stehen, diese keine komplexen Attribute haben, welche Referenzen benötigen würden und wir keine Offline Funktionalität brauchen. Falls die Datenbank jedoch über eine gewisse Größe hinweg wachsen würden, wäre ein Umstieg auf Cloud Firebase denkbar, da die Preispolitik diesen Usecase unterstützt.

Storage

Firebase bietet ein simples Filesystem, mit dem man Dateien in Ordnerstrukturen speichern kann. Von jeder Datei wird die Größe, der Typ, das Erstell- und Änderungsdatum und die Download URL angezeigt. Durch die SDK´s ist die Navigation durch die Ordnerstruktur und das downloaden von Dateien unkompliziert. Als Administrator erhält man eine Übersicht über alle Downloads, die momentane Speichergröße und die momentane Anzahl an Elementen, für einen bestimmten Zeitraum.

Hosting

Firebase ermöglicht es auch eine Webseite zu hosten. Eine Übersicht über alle Uploads mit deren User ist vorhanden um auch zwischen Versionen zu wechseln. Man kann seine Seite auch mit einer eigenen Domain verbinden.

Authentifizierung

Wie bereits erwähnt bietet Firebase hier viele Möglichkeiten für Entwickler Ihre User anzumelden. Anmeldeoptionen von etablierten Netzwerken wie Facebook, Twitter, GitHub, Google Play oder Google sind unterstützt, ebenso wie Telefonauthentifierung, Gastaccounts oder Authentifizierung über Email und Passwort. Es stehen auch Templates für Email-Adressen Bestätigung, Passwort zurücksetzen, usw. zur Verfügung.

Im Webinterface ist eine Auflistung aller registrierten Accounts, mit deren Anmeldeoption, dem Datum der Erstellung und dem Datum des letzten Logins, ersichtlich.

Werden Telefonbestätigungen verwendet, sieht man diese als Graph dargestellt. Die Einbindung der Authentifizierung in andere Firebaseprodukte ist über sogenannte Regeln implementiert. Hier kann der Admin einstellen ob, oder wer Schreibzugriff auf die Datenbank hat. Kann jeder schreiben, der eingeloggt ist, so gibt es die globale Variable „auth“, deren Wert man mit null vergleicht. Diese Regeln können mit einem Simulator getestet werden um Fehler zu vermeiden.

Preispolitik

Firebase bietet drei verschiedene Modelle, wobei eines kostenlos ist und die restlichen beiden kostenpflichtig sind.

Der „Spark Plan“ ist die gratis Version. Man hat Zugriff auf Datenbank, Fileserver, Website Hosting, Authentifizierung und den Großteil der restlichen Produkte. Jedoch sind bestimmte Nutzungsgrenzen für die verschiedenen Tätigkeiten festgelegt. Die Datenmenge der Realtime Database, darf zum Beispiel nicht über ein Gigabyte groß sein und es dürfen auch nicht über zehn Gigabyte heruntergeladen werden. Für Ludimus sind das jedoch astronomische Grenzen, die nur mit extrem vielen Spielen erreicht werden können. Unsere Anwendung verbraucht vielmehr Fileserverkapazitäten, die bei diesem Modell bei fünf Gigabyte liegen und es dürfen maximal ein Gigabyte pro Tag heruntergeladen werden. Mit genügend Nutzern sind das die ersten Schwellpunkte die überschritten werden.

Für 25$ pro Monat kann man Subscriber des „Flame Plans“ sein. Dieser hebt die Nutzungsgrenzen aller Produkte um einiges.

Da wir diese Erhöhung jedoch nur für ein Produkt brauchen, und wir hier nicht allein sind, ist die dritte Möglichkeit, der „Blaze Plan“, „Pay as you go“, heißt man zahlt nur, was man benötigt. Zur Schätzung der Kosten bietet Firebase hier einen Simulator und selbst wenn Ludimus viele Nutzer, mit vielen Spielen hätte, würde dieser Plan nur 20$ kosten. Zusätzlich dazu verfügt man dann jedoch auch über alle Funktionen, die Firebase bietet. Bigdata Analysen und mehrere Datenbanken- und Storage-Buckets für Parallelität sind nur einige davon.