Multi-Platform App Model for Distributed Education Software

Plurilogic

André Beauchamp, President & Founder
Sakis Spyrou, Project Manager
Topics of Discussion

- Introduction to Plurilogic and our Software
- Our Technology & Vision
- v.12 Defragmenter Tool Case Study
- Distributed Data Model for our Mobile App
- Quick Demo of the Plurilogic Mobile App
Introduction to Plurilogic

- Independent Software company specializing in education
- Founded in 1983 in Laval, Québec
- Currently 22 full-time employees and growing
- Leading school information system for private schools in Quebec.
- Our software has been implemented in more than 100 schools; roughly 200K users use our systems on daily basis.
- OEM partnership agreement with Actian since last year
- It’s our 3\textsuperscript{rd} participation at the Tech conference and we are committed to attend future events
The Plurilologic Integrated School Management Platform

An ERP for Schools

On-premise Desktop applications

Admin

FGA
FGB
FGF
FGP

Student Information
Lesson Planning
Communication

Finance & HR
After School & Daycare
Food Services

Admissions
Library
Analytics & Reporting

Teachers
Parents
Students
Admin

On-premise or on the cloud

pluriportail web + mobile
Our Technology & Vision
The Platform Architecture

- Database: ZEN
- Business Logic: C++ objects
- Application Server: SRF, HTML/JS
- Web Portal: Pluriportail
- Web services (JSON)
- Mobile App: JQuery / HTML5 Framework
- Office 365
- Windows Desktop Administration Tools: FGx
Software Distribution: Self-Serve Approach

Microsoft TFS Backlog + BUILD Server

3 weeks (sprint) minor Release

3 months Major Release

Release Package Installer .exe

School

On-premise

School

School

Cloud

School
ATL Server

- Fast DLL optimization
- No script to reinterpret
- Each line is compile
- Fast C++ access to databases
Why Zen

- Need a fast database
- Data Caching
- Able to respond to 3000 users simultaneously
- Support for 1:1 school programs
- Requiring little hardware resources
- Delivering rich multimedia content
- Optimizing Blob’s (Binary Large Object)
- Fast record locking allowing multiple transactions
Why not SQL:
A question asked by many customers

- MS does not use MS-SQL for Active Directory and Exchange
  - Probably the most important software for them
  - They are using Jet Engine
- OneDrive Enterprise uses MS-SQL and the overall functionality is not that great
Why not SQL: Data Security & Integrity

- We are in an open world.
  - Does all software need to be free?
  - Does all data need to be exposed to 3rd parties?
- Our Software is a grey box.
  - We give the information that our clients need without compromising integrity and security.
  - Examples: XLS export functions, Microsoft Data School Sync
- Our data structure is our secret sauce.
- It is more difficult to hide data structures with MS-SQL.
Why not SQL: Optimizing BLOBS

- Archiving everything
  - PDF
  - Emails
- Mail attachments
- Rich content
- Annotations
  - Hand writing
  - Voice
v.12 Defragmenter Case Study
v.12 Defragmenter Case Study

- Benchmark objective:
  Measure the Impact of the PSQL defragmentation tool on time consuming operations.

- Client:
  - Private Primary and Middle School in Montreal with 2,000 students
  - Plurilogic client for over 30 years

- Workstation environment:
  - Dell / Processor : Intel® Xeon® CPU E31220 @ 3.10GHz
  - Memory: 16 Gigs
v.12 Defragmenter Case Study

- Client Database
  - 338 Files
  - Database Size : 11.3 GB
  - PSQL vers. 12.00.160
Pre-defragmentation Operation:

- **Invoice purge** program:

Executed to optimize database storage, it deletes **invoices** and related records such as **invoice payments** that are older than 10 years.
In our particular example:

- School data contained 30 years worth of finance data
- Purge program deleted 235K invoices and 246K payment records
- As a result, the purge program created big gaps in the database file structure!
Pre-defragmentation Operation:

- Executed the **accounts receivable aging** program:

  Generates a report that lists unpaid customer invoices by date ranges.

**Pre-defrag elapsed time to execute report:**

~ 18m:49s
Defragmentation Details:

- Executed the defrag on the 5 database files of concern.
- Elapsed time for the defragmentation: ≤1 min
- Total storage size before the defragmentation: 1.6 GB
- Total storage size after the defragmentation: 485 MB
v.12 Defragmenter Case Study

Post-defragmentation Operation:

- Executed the accounts receivable aging report again

Post-defrag elapsed time to execute report: ~6m:14s

Execution time was reduced to a 1/3 of the wait time for the same report prior to running the defrag process!
v.12 Defragmenter Case Study

Other cases where our schools benefit from the PSQL defrag tool:

- Other data sets that we run purge operations for archiving:
  - Messaging

- Large data crunching operations:
  - Calculate marks / generate reports cards
  - Statistical Reports using historical data
Distributed Data Model for our Mobile App
**Pluriportfolio Mobile Technology Stack**

**1st generation** : launched 3 years ago
- Cross platform hybrid app (android, iOS & windows)
- Build using the Cordova application development framework
- HTML5/JQuery + Kendo UI components
- localForage in-browser data storage library for JS(caching)

**2nd generation** : release target Q3 of 2018
- migration to Angular
- Improve offline and data synchronization functionality
- Convergence of the web portal and mobile app; Share UI components and web services.
Offline First Experience For School Data

- **Improve Performance / Reduce Server Load**
  - Replicating server data on the mobile app will give more autonomy to the client; this should minimize server calls and reduce data transfer between the server and mobile app.

- **Support Complex Use Cases**
  - Connectivity to the server is not available (network disruption OR remote location).
  - Users with limited or no data plans (wi-fi only).

**Examples**
- Teachers need to take attendance on a school trip or at the school yard.
- Student heading to the cottage for the weekend without connectivity but needs access to homework;
On demand, one-shot synchronization of all data

Initialize data at the beginning of new school year.

Synchronization filtering based on user role or time spans.

Only synchronize email that spans a certain number of months.

Continuous synchronization of data anytime a change occurs.

Email should be available without any delays.

Incremental Synchronization of only data that has changed since the last sync.

Only pull email messages from the server that are new.
One-Way Synchronization

One-way Sync Type:
Data is only pushed from the server to the client.

Examples:
- List of students for a given classroom
- Student agenda
Two-Way Synchronization

Two-way Sync Type: Data is synced in two directions; from the client to the server and back.

Examples:
- Student tasks
- Attendance taking
Multiple-Way Synchronization

Multiple-way Sync Type:
Data is synced from multiple clients to the server and back.

It also implies that one client can sync to another via the server.

Examples:
- Chat between teachers and students
- Justification of an absence by both parents
Conflict Resolution

This implies making a choice between two versions of the same data object changed since the last sync.

- Multiple users change the same data object.
- Same user makes changes to the same data object from multiple devices or platforms (web portal).
- Stale Data. Users that stay offline for a long time may run the risk of working with outdated data.

Examples:

- Student sets task to '75% done’ on their tablet and then to DONE on their phone (both on offline mode).
- Both parents justify their child’s absence.
Option 1: Server priority (used in most cases)
- In the event of a conflict, server data overrides the client data object.
Conflict Resolution Strategies

Option 2: Timestamp priority (partially used)

In case of conflict, the data object version with the most recent timestamp wins.

1. Conflict detected

- 2A. Override client data with server data
- 3. Notify client that conflict resolution has taken place

- 2B. Override server data with client data

Server

ZEN (School Data)

Web services (JSON)

Client

Mobile App (phones, tablets)
Option 3: User priority (future consideration)

In case of conflict, the user is prompted to make a choice between the server and client data.

1. Conflict detected

2. User Makes Choice

A. Server Data Chosen
   3A. Override client data with server data

B. Client Data Chosen
   3B. Override server data with client data
Design Considerations and Decisions

- The client initiates a sync and not the server.
- Synchronization triggers can be user actions or automated triggers based on business rules.
- Clients must keep track of which data they have already synced and which not.
- Server logic handles conflicts and not the client.
Offline workflow

1. Sync request

2. add objects to the sync queue

3. sync queue with server

4. Check for Conflicts

5. Update Server

6a. Server Update succeeded (remove objects from sync queue)

6b. Server Update failed (set error codes)

7. Update Client Data
Building the 2nd Generation of Pluriportail Mobile with ZEN

- Implement transactional consistency via data locking (ex. Payment transactions).
- Integrated replication system that will simplify the process and minimize our synchronization code.
- Extend data replication to include secure data, BLOB data and parent/child dependencies.
- Improve performance and reliability using a 2 level data storage system.
## Mobile data storage system

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Mobile Browser Cache (localForage)</th>
<th>Cache storage for all school data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 2</td>
<td>Persistent Storage (ZEN)</td>
<td>None</td>
</tr>
</tbody>
</table>

1<sup>st</sup> generation of Pluriportail Mobile (single level data caching)
# Mobile data storage system

<table>
<thead>
<tr>
<th><strong>Level 1</strong></th>
<th><strong>Level 2</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Browser Cache (localForage)</td>
<td>Persistent Storage (ZEN)</td>
</tr>
</tbody>
</table>

### 2nd Generation of Pluriportail Mobile
(2-level persistent data storage with caching)

- Limit the cache to high-use, non-sensitive school data for fast access

### Level 2 Persistent Storage (ZEN)

- Store sensitive data (Authentication, sync queue)
- Allow us to replicate more data (BLOBS)
- Failover Mechanism
- Execution of complex SQL-like queries
Quick Demo of Pluriportail Mobile