



# Acumatica Developer Roadmap

Mike Chtchelkonogov  
Founder & Chief Technology Officer  
Acumatica

# Agenda

---

- Core Platform Priorities
- Work-in-Progress
- The Future – 2021 & Beyond
- Questions & Answers

## Core Platform Priorities

# Core Platform Priorities

---

- ✓ Continuous platform technology advances
- ✓ Improved UI and usability
- ✓ Non-programming customization
- ✓ Dashboard and reporting enhancements
- ✓ Machine learning and artificial intelligence
- ✓ Performance and scalability





## Work in Progress (2020-2021)

# Replacement of the Web.Forms Technology

---

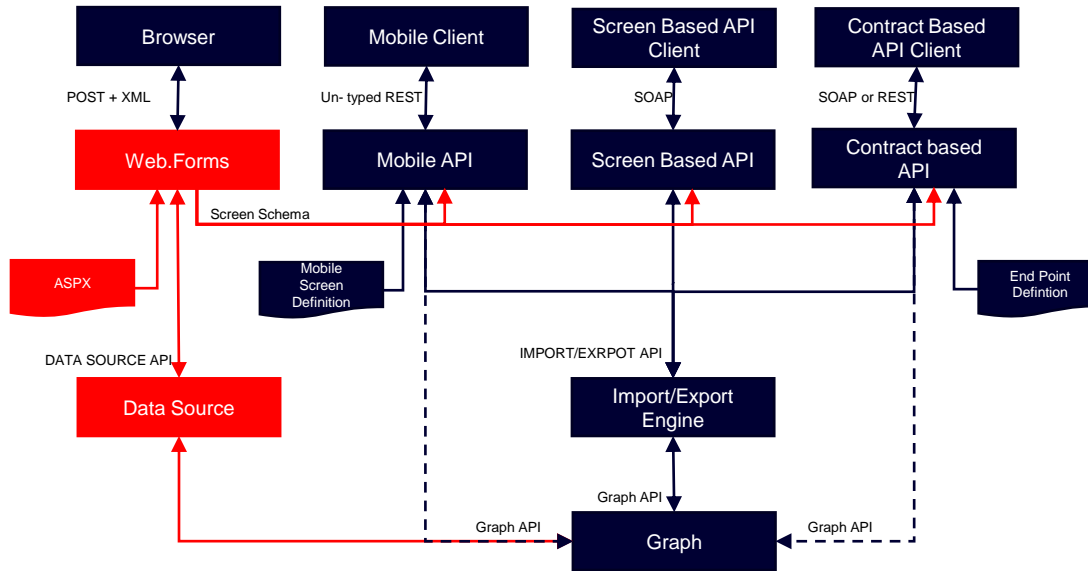
## What...

- Replace the legacy Web.Forms technology with a new template-based browser-based engine.
- Unifying API for the Web & Mobile clients.
- Migrate existing web forms and customizations to the new technology.

## Why...

- Eliminate legacy technology and implement support for .NET Core
- Improve application performance
- Unify the communication layer API
- Enable custom visualizations for the 3<sup>rd</sup>-parties

# Current Frontend Architecture



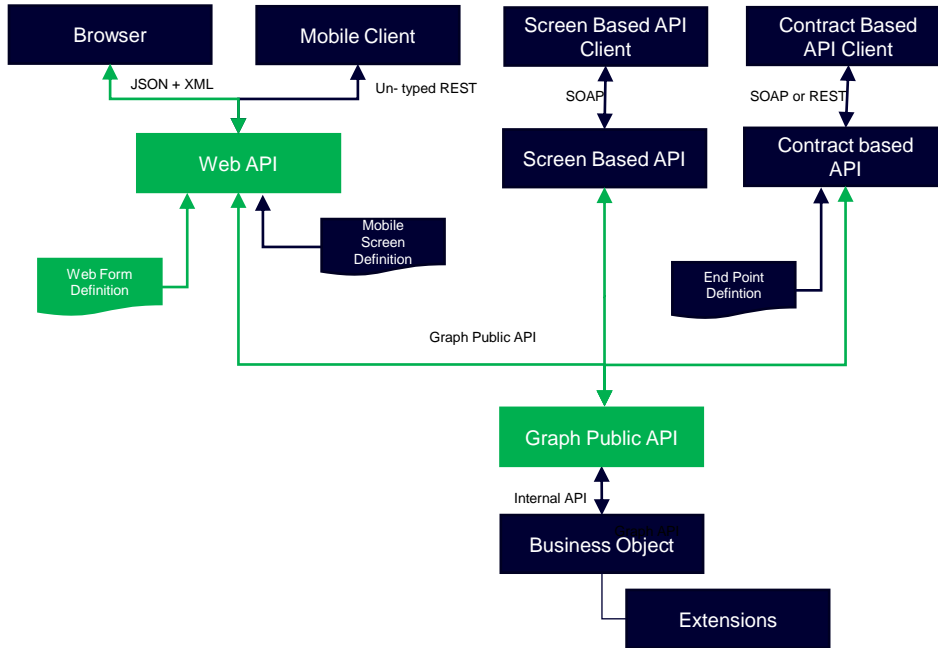
Web.Forms technology is legacy and going to be obsolete in 2-3 years

Web.Forms technology is heavy; it consume up to 30% of CPU time and request execution time

Datasource and Import/Export Engine provide two alternative options to access API

Meta data generated out of ASPX form is used for other API's

# Future Frontend Architecture



Web.Forms technology replaced with modern web API technology

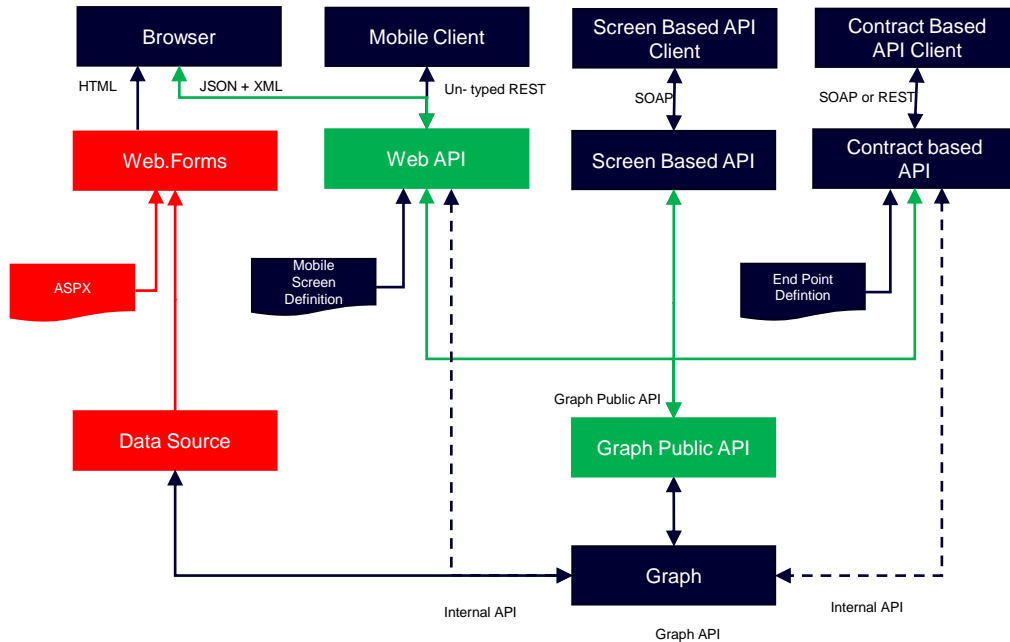
Mobile and Web API are uniform and can be unified as a single component

Public graph API does not depend on UI definition and uniform for all external access introducing a public contract

Public graph API creates a point for implementing simplified integration tests against the business object



# Frontend Architecture – Migration Path



Step 1 – Modify Mobile API to handle HTTP requests from browser and modify JS to bypass web forms and work through Web API.

Step 1 – Work on Import/Export Engine to expose public graph API that will be uniform for all frontend engines.

Step 2 – Replace Web Forms with new rendering engine based on aurelia.io

Step 3 – Convert old forms to the new format

# Match Mobile UI functionality to Browser

---

## What...

- Implement Workspaces support on mobile devices
- Implement support for secondary containers (adding lines to the grid)
- Implement popup support

## Why...

- With more mobile usage, we need to optimize the mobile interface for more efficient data entry and data access. Right now, some of the functions available in the browser are still superior to the mobile application and the mobile application does not provide the same experience as the browser.
- Some operations on mobile work different on the mobile interface than on the browser due to the form factor. Adoption is required for these functions to provide a similar user experience.

# New State Automation Engine

---

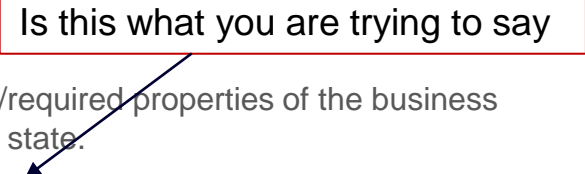
## What...

- Setting the properties of the business object based on the evaluation of the object state
- Gradual completion of the business object fields based on the object state
- Triggering the business events based on the object state
- Enabling and disabling the actions on the business object based on the object state
- Defaulting and restricting the field values based on the other field values
- Configurable in the code with the base object and extendable in customization

## Why...

- 80% of the UI programming or customization is managing enabled/disabled/required properties of the business object fields and actions depending on the entered information or the object state.
- Support of these actions through the state automation engine **will not involve programmers** for customizing the form behavior which will reduce the qualification requirements to perform customization and reduce the customization cost.

Is this what you are trying to say



# Extending User Defined Fields fields functionality

---

## What...

- Support of lookup functionality for the UDF field definition
- Managing UDF fields properties by means of state automation
- Defaulting and restricting UDF fields by means of state automation
- Support for copy of UDF field values during a new record creation
- Configuration of the different UDF fields sets for the different types of record

## Why...

- UDF Fields are the simplest way to extend the business logic and it is very stable on upgrades.
- Extending this functions will extend use of these fields for basic customization will reduce the need for a programmer and will speed up customization and the implementation cycle.

# Improve Generic Inquiry Usability and Functionality

---

## What...

- Automatically suggest links when designing inquiry
- SQL like syntaxis for advanced users
- Use GI on top of GI
- Improve UI for GI configuration
- Add charts
- Implement subtotals
- Improve filtering and tabs behavior.
- Implement report designer on top of GI

## Why...

- Reporting is the core function of the ERP solution and the better the reporting tools are, the better the ERP
- Need to convert GI to the full functioning online reporting tool

# Replacement of the legacy notification mechanism with the Business Events

---

## What...

- Implement support for the quick notification configuration using Business Events
- Implement self subscription function to subscribe to watch for record changes

## Why...

- This option will give a user an option to be notified about the changes on a specific entity or about a specific event through notification mechanisms.
- Right now, to configure a notification on top of a business event the user must configure the push notification inquiry for this event, configure the event, and then link the notification through the event handler. This is complicated for the user compared to the legacy notification mechanism and prevents us from dropping it.

# Improve system troubleshooting

---

## What...

- Provide a central screen to view the system events and errors that occurred during unattended operations like scheduled processes or detected system performance or stability issues.
- Provide a central notification mechanism for the administrator to be notified on such events.

## Why...

- Right now, detections and troubleshooting of these events is complicated and results in an excessive support payload.
- Additionally, it is often considered a distraction by clients.



## The Future – 2021 & Beyond



# Scripting Customization Layer

---

## What...

- Implement a new customization layer between the UI and business logic, where alterations to the business logic can be implemented by means of a basic scripting language using the public business object API

## Why...

- Security isolation of customization from code
- A more simplistic language that lowers entry barrier and can be learned by consultants
- Higher customization stability on upgrades due to the use of a public API vs the internal programming API by custom code

# Replace ARM with Pivots In Memory Engine

---

## What...

- Extend the Pivot engine functionality to replace the existing ARM engine
- Eliminate historical records and use pre-aggregated transactional data

## Why...

- Improve system throughput by eliminating the locks on historical tables during the release
- Improve system responsiveness by storing the pre-aggregated data required for the reporting and appending them with transactional data during the report execution
- Provide daily balances in ARM

# Dimensional Analysis

---

## What...

- Option to represent a subaccount as a set of independent or inter-related controls during data entry
- Option to indicate what segment is and is not required and hide any controls not required
- Option to select multiple values of these segments in reports and inquiries
- Modify the configuration screens to configure the dimensions in a user-friendly form

## Why...

- Dimensions are more convenient than subaccounts from the user's perspective

# On-screen Content Help

---

## What...

- Implement context help for screen elements
- Eliminate the complex screen description context help that explains the screen elements in place

## Why...

- Currently discovering the meaning of functions of a specific screen element is cumbersome and requires additional efforts. Implementing context help will shorten the education cycle and as a result, will accelerate system implementation and adoption by end-users

# Moving Acumatica to .NET Core

---

## What...

- Move Acumatica libraries to .NET core technology
- Make Acumatica run on .NET Core inside Docker containers

## Why...

- Remove dependencies from the legacy components of the .NET Framework
- Run on Linux
- Better performance and application density

Questions?



# Thank You

---

**Mike Chtchelkonogov**