Thomson Reuters™

TR-Contabilidad modernization proposal

The plan proposes a gradual migration to a modern architecture in .NET 8/9 technology, designed to be secure, flexible, capable of running on premise but cloud-ready, and without interrupting the current operation of customers. The strategy will be iterative and controlled, reducing both technical and human risks.

In addition, modernization incorporates best practices such as feature-based decoupled architecture, continuous integration and delivery (CI/CD), automated testing, and silent update mechanisms. This will optimize support costs, improve the user experience, and ensure the technological evolution of the solution.

Current Situation

Aspect	Description
Technology	Desktop application developed in **Visual Basic 6**. Source code contains approximately 250 file forms (`.frm`), components, and dependencies. It is distributed as an installer and requires an additional executable for updates.
Modules	**Module 1 (Administrator):** manages companies and transversal data. **Module 2 (Client):** manages current accounts, fixed assets, legal books, and article 14D taxes.
Database	Local databases in **Microsoft Access** or **SQL Server Express** installed **on-premise**.
Users	Some use old versions of Windows, which limits the ability to update components.
Problems	Dependency on VB6 (obsolete platform without support), low compatibility with current environments, installation difficulties on modern systems, costly maintenance, dependency on external updater.

Risks of Current State

- **Limited support and security risks:** VB6 has vulnerabilities and high maintenance costs.

- **Low modularity:** the code is distributed across many forms and modules with dependencies.
- **Dependence on outdated operating systems:** Some customers use old versions, limiting adoption of new technologies.

Desired Situation

- Use modern technology.
- Keep the application on-premise, but prepared for cloud scenarios.
- Minimize external software dependencies on client machines as much as possible.
- Organize code in a decoupled, feature-based, cloud-ready structure.
- Modernize the user interface to improve user experience.
- Include automatic updates with rollback option.
- Comply with the company's security and compliance policies.

Technologies to Consider:

- **Language and runtime:** C#.NET and .NET 9 as a basis for modern applications, self-contained or framework-dependent.
- **Database:** SQLite/SQL Server LocalDB depending on the scenario.
- **Feature-based and cloud-ready architecture:** the code will be organized by features, encapsulating commands, queries, business logic, data access, and controllers in each feature. This minimizes coupling and improves cohesion. Code prepared for possible cloud migration.
- **Modernized interface:** the presentation layer could be implemented in **MAUI** for cross-platform compatibility, **WPF**/WinForms self-contained, or **Blazor WebView** for a local web approach. A **REST API backend** in ASP.NET Core will decouple logic and facilitate a potential cloud migration.
- **Automatic updates:** the app will be distributed via MSIX/WiX Toolset/Squirrel installers or self-contained executables, and will include an internal service that checks for new versions and updates silently.
- **CI/CD:** CI/CD pipelines will be configured in GitHub Actions to build, test, and publish each release.
- **Unit and integration tests:** adopt testing from the start and use AI tools to generate documentation and test cases.
- **Team engagement and motivation:** foster a culture of **continuous improvement** and learning. Show the benefits of the new approach and the opportunity to use new technologies.

Strategy by Stages

1. **Analysis and Planning**
   - **Code inventory**: evaluate VB6 code, external components, and dependencies. Classify modules by criticality and difficulty.
   - **User and support team interviews**: identify critical features and workflows. Involve users to get their perspective.
   - **Technology evaluation**: perform proof-of-concepts with .NET 9/8 to validate compatibility and performance. If self-contained application size is problematic, consider publishing them as *framework-dependent* (lighter installer) or using *trim* techniques.
2. **Design of Desired Architecture**
   - **Backend API**: implement a REST API with ASP.NET Core organized into vertical slices (e.g., `Features/Companies`, `Features/Assets`, etc.). Each slice will contain commands, queries, validations, services, and controllers.
   - **Data layer**: use **Entity Framework Core** for persistence. Models can be generated from the current database, adapting data types.
   - **Presentation layer**: choose WPF/MAUI/Blazor WebView according to user experience and hardware requirements. The UI will consume the local REST API. It is recommended to involve the product team to collaborate on mockups.
   - **Update services**: integrate a module that checks versions on a central server and performs silent updates. For customers without permanent connectivity, consider offering an installer.
3. **Refactoring and Decoupling**
   - **Incremental strategy**: start with the creation of new features in the modern architecture and wrap the existing VB6 system. Gradually migrate VB6 modules to C# and retire old code. This avoids the *big bang* approach and reduces risk.
   - **Shared services**: extract utilities and common logic into independent libraries to avoid duplication.
   - **Automated tests**: as each module is migrated, create unit tests comparing results with the VB6 version to ensure equivalence.
4. **AI Tools and Automation**
   - **VB6 code analysis**: use Generative AI tools to map dependencies, generate documentation, and suggest translations to C#. This can reduce modernization times and improve code understanding.
   - **Test generation**: use AI to create test cases and documentation. Automation of tests and documentation reduces effort and risk of errors.
   - **Copilots and assistants**: integrate tools like GitHub Copilot for refactoring and design pattern adoption.
5. **Change Management and Developer Motivation**

- **Culture of continuous improvement**: promote the mindset that change is evolutionary, not disruptive. Communicate that modernization offers learning and professional growth opportunities.
- **Recognition and participation**: recognize team contributions and assign responsibilities in design decisions. Listen and demonstrate benefits with proof-of-concepts.

6. **Migration and Deployment Strategies**
   - **Iterative migration**: move functionalities module by module. Each iteration will include analysis, development, testing, and pilot deployment.
   - **Parallelism**: allow VB6 and .NET versions to coexist for a period to validate results.
   - **User testing and feedback**: release beta versions to selected users to gather early feedback.
   - **Final deployment plan**: once most features are migrated, plan the general release. Provide support for incidents.

Estimates

The following backlog is designed considering **only one person** and uses estimates in hours (1 day = 7 hours). Three scenarios are considered: Optimistic (O), Normal (N), and Pessimistic (P). Parallelizable tasks are also identified.

Feature / User Story	Task Summary	O (h)	N (h)	P (h)	Possible Parallelism
Feature 1: Diagnosis and Planning					
As an analyst I want to inventory VB6 code	Run analysis tools, catalog forms and dependencies, document risks.	28	42	70	Sequential
As a product owner I want to interview users and support team	Plan interviews, hold meetings, gather feedback and expectations.	28	42	56	Sequential
As a product owner I want to define the vision of the desired situation	Set goals, prioritize features, create roadmap.	14	21	35	Sequential

Feature / User Story	Task Summary	O (h)	N (h)	P (h)	Possible Parallelism
Feature 2: Design and Preparation					
As an architect I want to design the vertical slice architecture	Create folder structure by feature, define dependencies, document coding standards.	21	35	56	Parallel with other preparation tasks
As a DBA I want to design the SQLite schema and migration strategies from Access/SQL Express	Define data model, write migration scripts, validate integrity.	35	49	84	Parallel with other preparation tasks
As a dev I want to configure CI/CD with GitHub Actions	Create workflows for build, test, static analysis, and packaging.	21	35	49	Parallel with other preparation tasks
As a dev I want to configure AI tools	Select tools, integrate them into the repo, document usage.	14	21	35	Parallel with other preparation tasks
As a product owner I want to define training and motivation strategies	Design training plan, coaching, and communities of practice; prepare communication sessions.	21	35	49	Parallel with other preparation tasks
Feature 3: Iterative Migration (per module)					
As a dev I want to migrate the Administrator module to C#/.NET	Design API and UI for company management; migrate logic; create tests; deploy pilot version.	105	140	210	Sequential

Feature / User Story	Task Summary	O (h)	N (h)	P (h)	Possible Parallelism
As a dev I want to migrate the Current Accounts module to C#/.NET	Implement `CurrentAccounts` slice, rewrite calculations, tests, deployment.	105	140	210	Sequential
As a dev I want to migrate the Fixed Assets module to C#/.NET	Migrate depreciation calculations, reports; implement corresponding slice.	84	112	175	Sequential
As a dev I want to migrate the Legal Books and Taxes module to C#/.NET	Migrate generation of books and article 14D taxes; validate calculations.	105	140	210	Sequential
As a dev I want to decouple business logic from the UI by implementing REST API and services	Create commands and queries, separate domain rules, expose endpoints.	56	84	126	Sequential
As a dev I want to integrate automatic update service into the new application	Design check, download, and installation mechanism.	35	49	84	Sequential
As a dev I want to package the application into self-contained or framework-dependent installers depending on size and support analysis	Configure MSIX or executables, sign certificates, validate on clients.	21	35	56	Sequential
Feature 4: Testing and Stabilization					

Feature / User Story	Task Summary	O (h)	N (h)	P (h)	Possible Parallelism
As a tester I want to run unit and integration tests to ensure parity with VB6	Configure test projects, write cases, and compare results.	35	49	70	Parallel with other testing tasks
As a key user I want to validate the migrated application to ensure it meets needs	Plan pilot testing, collect feedback, report issues.	28	42	56	Parallel with other testing tasks
As a dev I want to optimize performance and fix errors detected during testing	Analyze performance reports, refactor, and adjust queries.	21	35	56	Parallel with other testing tasks
As a product owner I want to prepare manuals and training material for end users	Create documentation, training videos, adoption plans.	28	42	56	Parallel with other testing tasks
As a product owner I want to plan final deployment to clients securely	Define installation windows, communication, contingencies, and support.	21	35	49	Parallel with other testing tasks
Feature 5: Support and Continuous Improvement					
As a dev I want to handle incidents and release fixes after launch	Maintain ticket system, assign priorities, release patches.	14	28	42	Sequential
As an architect I want to explore backend cloud migration in the future	Evaluate platforms, analyze costs, create prototypes.	35	56	84	Sequential
As a product owner I want to measure usage and	Implement telemetry, analyze feedback, adjust	21	35	49	Sequential

Feature / User Story	Task Summary	O (h)	N (h)	P (h)	Possible Parallelism
satisfaction metrics to guide improvements	backlog.				
Total		896	1302	1967	

Considerations and Risks

- **Complexity of third-party components:** identify unsupported VB6 controls or libraries and evaluate modern alternatives or develop replacements.
- **Size and updates of self-contained applications:** self-contained apps include the runtime and all libraries, so their size and storage requirements are larger. Evaluate if a *framework-dependent* deployment (requires installing .NET on the client) reduces size and simplifies updates.
- **Data management:** when migrating to SQLite, consider limitations (concurrency, size) and implications of migrating data from Access/SQL Server. Make backups and run integrity tests.
- **Incremental planning:** avoid a *big bang* approach and adopt gradual migration to minimize risks.

Benefits

- **Reduced support costs:** fewer issues in installation and compatibility.
- **Increased user productivity:** modern interface, faster response times.
- **Regulatory compliance:** mitigation of legal and compliance risks.
- **Technical scalability:** the solution is prepared to support new tax regulations or business changes with localized adjustments, avoiding massive rewrites.
- **Lower risk of regressions:** each module is encapsulated and independently tested.
- **Ease of implementing new features:** thanks to feature-based decoupled architecture, functionalities can be added or modified quickly with low impact on the rest of the system.

Conclusion

This plan not only modernizes the accounting application, but also establishes a scalable technological foundation. The decoupled, cloud-ready architecture ensures that the investment will not become obsolete, while the iterative strategy and change management reduce technical and human risks. Thus, TR-Accounting not only ensures operational continuity, but becomes a flexible platform, ready to quickly respond to new regulatory, technological, and customer needs.