ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК 004.05653
DOI: 10.21822/2073-6185-2022-49-2-33-45 Оригинальная статья /Original Paper

Алгоритм применения кроссплатформенного мобильного приложения для защиты информации в облачном хранилище

А.Р. Газизов
Донской государственный технический университет,
344002, г. Ростов-на-Дону, пл. Гагарина, 1, Россия

Резюме. Цель. Качественное «облачное хранилище» данных должно обеспечивать конфиденциальность, целостность и доступность информации в «хранилище». Использование «облачных хранилищ» для хранения информации в резервных копиях сопряжено с их возможной утратой. Обеспечение конфиденциальности информации в «облачном хранилище» предполагает, что, «выгружая» данные в «облачное хранилище», пользователь должен быть уверен, что злоумышленник не сможет получить информацию из хранилища. Целью исследования является разработка алгоритма применения кроссплатформенного мобильного приложения для защиты информации в «облачном хранилище» ИС. **Метод.** Исследование основано на использовании кроссплатформенного подхода создания приложений. **Результат.** Проведен анализ существующих подходов и фреймворков разработки кроссплатформенных мобильных приложений. Предложено создавать приложение на основе фреймворка Xamarin. Для обеспечения конфиденциальности информации предлагается перед выгрузкой данных и резервных копий файлов шифровать их. Шифрование предполагает сокрытие исходного вида файла от неавторизованного использования при помощи крипто-алгоритмов, когда авторизованным пользователем будет являться обладатель ключа шифрования. Для реализации функции шифрования данных и резервных копий файлов, выгружаемых в «облачное хранилище», предлагается к применению симметричный алгоритм блочного шифрования Rijndael, принятый в качестве стандарта шифрования по результатам конкурса Advanced Encryption Standard (США). **Вывод.** Кроссплатформенное мобильное приложение может компилироваться и эффективно работать под управлением операционных систем iOS и Android, выполнять шифрование файлов и выгружать их в «облачное хранилище» с возможностью последующей дешифровки.

Ключевые слова: доступность информации; информационная безопасность; информационная система; конфиденциальность информации; облачное хранилище; облачные технологии; пользователь информации; провайдер облачных услуг; целостность информации; шифрование информации.

Для цитирования: А.Р. Газизов. Алгоритм применения кроссплатформенного мобильного приложения для защиты информации в облачном хранилище. Вестник Дагестанского государственного технического университета. Технические науки. 2022; 49(2):33-45. DOI:10.21822/2073-6185-2022-49-2-33-45

An algorithm for using a cross-platform mobile application to protect information in cloud storage

A.R. Gazizov
Don State Technical University,
1 Gagarin Square, Rostov-on-Don 344000, Russia

Abstract. Objective. A high-quality "cloud storage" of data should ensure the confidentiality, integrity and availability of information in the "storage". The use of "cloud storage" for storing information in backups is associated with their possible loss. Ensuring the confidentiality of information in

the "cloud storage" implies that by "uploading" data to the "cloud storage", the user must be sure that an attacker will not be able to obtain information from the storage. The aim of the study is to develop an algorithm for using a cross-platform mobile application to protect information in the "cloud storage" of IS. **Method.** The study is based on the use of a cross-platform approach to creating applications. **Result.** The analysis of existing approaches and frameworks for the development of cross-platform mobile applications was carried out. It is proposed to create an application based on the Xamarin framework. To ensure the confidentiality of information, it is proposed to encrypt them before uploading data and backup copies of files. Encryption involves hiding the original type of the file from unauthorized use using crypto-algorithms, when the owner of the encryption key is the authorized user. To implement the function of encrypting data and backup copies of files uploaded to the "cloud storage", it is proposed to use the Rijndael symmetric block encryption algorithm, adopted as an encryption standard according to the results of the Advanced Encryption Standard (USA) competition. **Conclusion.** A cross-platform mobile application can be compiled and run efficiently under iOS and Android operating systems, encrypt files and upload them to the "cloud storage" with the possibility of subsequent decryption.

Keywords: information availability; information security; information system; information confidentiality; cloud storage; cloud technologies; information user; cloud service provider; information integrity; information encryption

For citation: A.R. Gazizov. An algorithm for using a cross-platform mobile application to protect information in cloud storage. Herald of the Daghestan State Technical University. Technical Science. 2022; 49 (2): 33-45. DOI: 10.21822 /2073-6185-2022-49-2-33-45

Введение. На сегодняшний день существует множество различных фреймворков и сред разработки, на которых можно успешно разрабатывать кроссплатформенные мобильные приложения. Однако следует определить, почему именно кроссплатформенный подход представляется наиболее приемлемым [1,4,5].

Противоположностью кроссплатформенному подходу является нативный подход, представляющий разработку приложения под конкретную платформу, используя определенные инструменты. Из преимуществ данного подхода следует выделить увеличенную производительность приложения, меньший размер и прочее. При этом в сравнении с кроссплатформенным подходом выделяются недостатки, связанные со сложностью разработки и совместимостью приложения; по причине того, приложения, написанные под определенную операционную систему, не будут совместимы с другими. Таким образом, разработчикам следует разделять человеческие ресурсы для параллельного создания различных версий приложения под разные операционные системы.

Постановка задачи. Решение проблемы при этом очевидно – использование кроссплатформенного подхода создания приложений, т.к. при его использовании исключается необходимость разработки различных версий приложения, ведь исходный код в итоге компилируется под разные платформы [1,2,3,6].

При этом важным этапом разработки является выбор инструмента разработки, т.к. каждый из них имеет свои особенности; инструменты могут различаться функциональностью, а так же языком программирования, на котором и происходит разработка.

Методы исследования. Алгоритм применения кроссплатформенного мобильного приложения. К основному преимуществу кроссплатформенного подхода следует отнести сокращение периода разработки приложения; что позволит в «больших» проектах сэкономить значительную часть бюджета, а также так же сократить затраты человеко-часов [10,11].

До недавнего времени нативный подход был единственным подходом к разработке приложений из-за того, что применение кроссплатформенного подхода сопровождалось значительным количеством ошибок и трудностей, возникавших в процессе разработки. Однако, ми-

ровые IT-гиганты (Google, Microsoft, Adobe и пр.) начали поддерживать данное направление, что существенно повлияло на развитие и распространение кроссплатформенного подхода, сделав его простым и эффективным методом разработки приложений под различные платформы [7].

На основе статистических данных портала Statista, денежный оборот приложений, созданных с помощью инструментов кроссплатформенной разработки, будет превышать 10 млрд. долларов США [8]. В настоящее время существует множество инструментов, позволяющих выполнять кроссплатформенную разработку мобильных приложений. Рассмотрим наиболее популярные из них (рис. 1) [11].

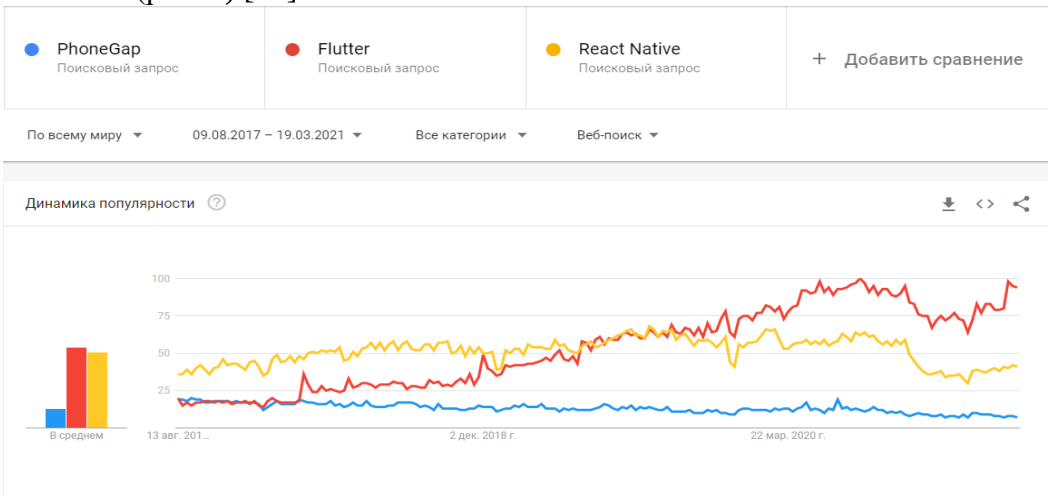

Рис.1. Статистика сайта Google по поисковым запросам PhoneGap, Flutter, React Native
Fig.1. Google site statistics for search queries PhoneGap, Flutter, React Native

PhoneGap (рис. 2). Adobe PhoneGap – это стандартизированная среда разработки с открытым исходным кодом для создания кроссплатформенных мобильных приложений с использованием HTML, CSS и JavaScript для iOS.

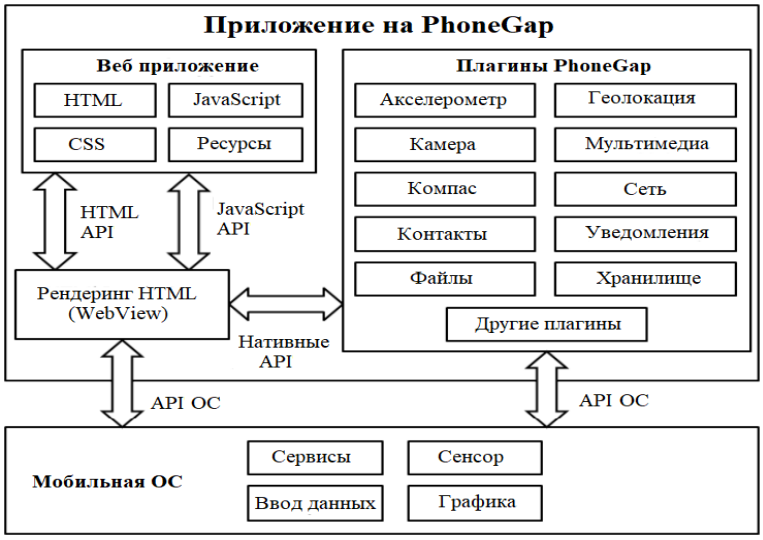

Рис. 2. Структура компонентов Adobe PhoneGap
Fig. 2. The structure of Adobe PhoneGap components

Данная среда разработки не требует специализированного аппаратного обеспечения, компиляторов, а также предполагает низкие системные требования, что делает ее эффективной и доступной.

Особенностью данного инструмента является то, что итогом разработки приложения являются классические HTML страницы, которые впоследствии открываются встроенным в мобильное устройство браузером. Интерактивность и создание пользовательского интерфейса

ложится на плечи JavaScript, который связывает созданные разработчиком функции с элементами DOM и аппаратными частями конечного устройства пользователя.

Adobe PhoneGap является наиболее удобным вариантом, если необходимо создать легкое и быстрое приложение с простым интерфейсом, либо WebView сайта [1-7].

React Native (рис. 3). Также, как и Adobe PhoneGap, React Native представляет собой инструмент с исходным кодом для разработки кроссплатформенных мобильных приложений. Для написания приложения могут использоваться как чистый JavaScript, так и его библиотеки (TypeScript и ReactJS). Знание и применение библиотек значительно упрощает и ускоряет разработку из-за имеющихся в них готовых функций и методов.

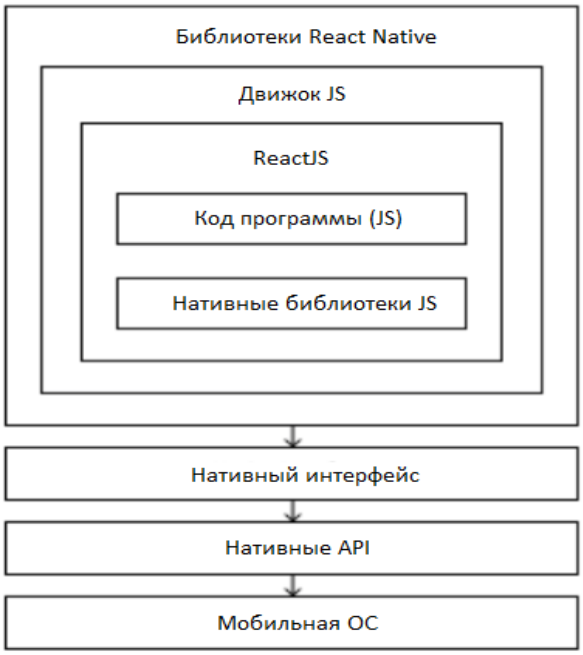

Рис. 3. Структура компонентов React Native
Fig. 3. Structure of React Native Components

React Native разрабатывается Facebook, Inc. и на сегодняшний день применяется для написания различных приложений – от простых (одностраничных) вариантов, до полноценных мобильных платформ разного рода. Из-за того, что готовое приложение открывается встроенным браузером устройства, то сохраняется быстродействие приложение сравнимое с нативным. При этом React Native поддерживает горячую перезагрузку; это означает, что внесенные в код изменения сразу же отобразятся на тестируемом устройстве, либо эмуляторе [1-7].

Flutter (рис. 4). Слоган инструмента кроссплатформенной разработки Flutter – «Красивые приложения в рекордное время». Flutter выпускается и разрабатывается компанией Google и представляет набор инструментов для создания легких и оптимизированных приложений с единой кодовой базой, которая впоследствии компилируется и предстает для конечной платформы в качестве нативного приложения.

Разработка приложения с инструментом Flutter по сравнению со многими другими подобными инструментами не происходит на языке JavaScript и его производных, в чем и заключается его основная особенность. В качестве основного языка разработки выступает Dart, который в процессе компиляции транслируется в двоичный код.

Таким образом, достигается «та самая» нативная производительность. Flutter включает в себя виджеты, которые оптимизируют процесс разработки, делают его более простым и понятным. Данные виджеты позволяют удобно выстраивать элементы пользовательского интерфейса, а также создавать различные 2D анимации. Готовые приложения занимают на конечном устройстве небольшое пространство, относительно других подобных инструментов кроссплатформенной разработки. Для отрисовки графики Google разработала собственный «движок»

Skia, написанный на языке C++. Он позволяет обрабатывать графические элементы, компилируя их для конечной системы [1-7].

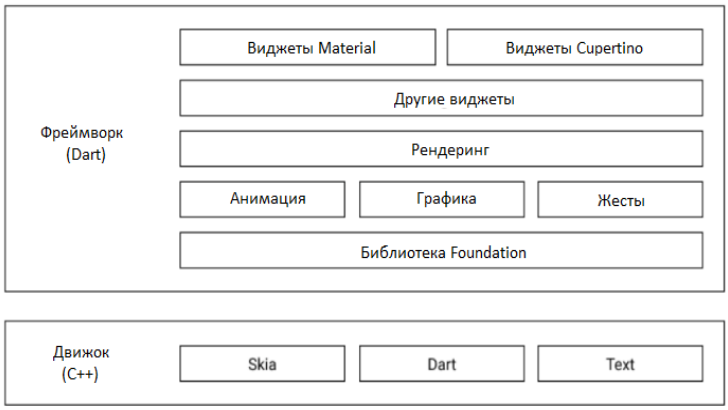

Рис. 4. Архитектура Flutter
Fig. 4. Flutter architecture

Xamarin.Forms (рис. 5). Xamarin Forms является «продукцией» компании Microsoft и распространяется через среду разработки Microsoft Visual Studio. Представляет собой инструмент кроссплатформенной разработки мобильных и дектопных приложений с использованием языка программирования C# и языка разметки XAML.

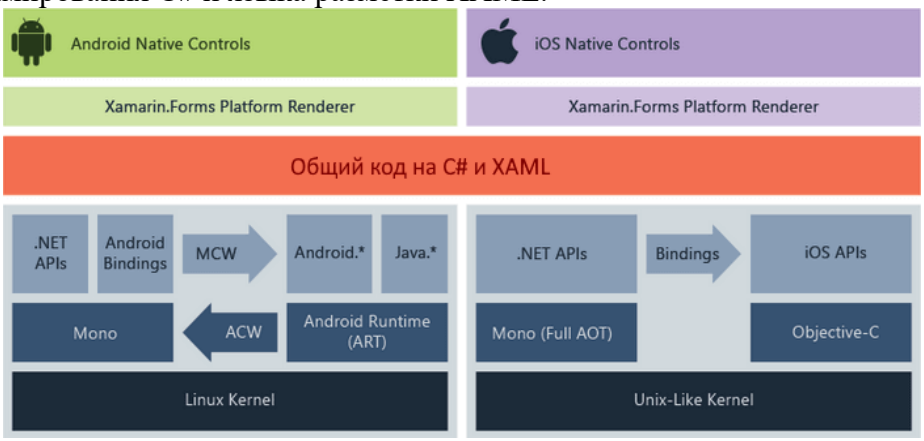

Рис.5. Компоненты инструмента Xamarin.Forms
Fig.5. Xamarin.Forms Tool Components

Использование Xamarin.Forms предполагает создание приложений под платформы iOS, Android, а также UWP, которые включают в себя Windows 10 и 8 (как на архитектуре x86, так и на архитектуре ARM), а также системы семейства Windows Phone 8 версии и старше.

Использование Xamarin является бесплатным и распространяется по лицензии MS Visual Studio. Начиная с пятой версии Xamarin.Forms появилась встроенная библиотека Xamarin.Essentials, которая облегчает доступ к аппаратным и программным компонентам любого поддерживаемого конечного устройства. С помощью этой библиотеки приложение может использовать в своей работе файловую систему устройства, акселерометр, телефон, воспринимать речь, а также управлять блокировкой устройства, если конечно пользователь даст на это права.

Также Xamarin.Essentials включает в себя собственную оболочку для пользовательского интерфейса, что многократно сокращает затраты времени на ее написание. 5 января 2021 года вышла последняя версия данной платформы – Xamarin Forms 5.0 и, собственно, она и будет рассматриваться.

Компиляция языка C# в нативный код для конечного устройства происходит благодаря фреймворку Mono, который является реализацией библиотек .NET Framework с открытым исходным кодом [1-7].

У Xamarin.Forms есть четкая структура, каждый элемент которой отвечает за последующую компиляцию кода:

1) Xamarin.Android – библиотека, реализующая компиляцию исходного кода для платформы Android

2) Xamarin.iOS – библиотека, реализующая компиляцию исходного кода для платформы iOS

3) Xamarin.UWP – библиотека, реализующая компиляцию исходного кода для Windows 10 и 8 (как на архитектуре x86, так и на архитектуре ARM), а также системы Windows Phone.

Эти библиотеки отвечают за обращение к компонентам конечного устройства – через них приложения могут направлять запросы к API устройства, а также к аппаратным компонентам Android или iOS.

При компиляции приложения под Androind код C# при помощи библиотеки Xamarin.Android компилируется в, так называемый, «высокоуровневый ассемблер», который в последствии переходит в нативную сборку системы.

Для запуска приложений в среде ОС Android используется Mono, который не может напрямую обращаться к программным компонентам операционной системы и аппаратной части устройства, однако для осуществления этих операций происходит обращение к пространствам имен Java. Таким образом, запросы от Mono могут транслироваться в нативные вызовы и, тем самым, обращаться к недосягаемым иным путем компонентам.

Приложения на iOS в отличие приложений на Android, который применяет для работы «приложения компиляцию на лету», используют технологию компиляции перед исполнением. Xamarin использует специальные запросы, которые именуются селекторами и регистраторами, которые представляют собой промежуточный слой между приложением на Xamarin и операционной системой на Objective-C.

Xamarin.Forms объединяет Xamarin.Android и Xamarin.iOS, тем самым, организуя общую кодовую базу, которая впоследствии компилируется для необходимой платформы. Таким образом, имеется возможность создавать единый пользовательский интерфейс на языке разметки XAML, к которому привязана некоторая логика на C#. Затем с помощью специфичных для каждой платформы визуальных интерпретаторов интерфейс будет преобразован с учетом его нативной стилизации [1-7].

Основные преимущества использования инструмента Xamarin.Forms:

1) Общий пакет технологий для всех платформ.

Xamarin использует язык C#, XAML и Mono, для разработки приложения под наиболее популярные на сегодняшний день платформы. Из-за единой кодовой базы имеется возможность повторно использовать до 80% исходного кода. Все приложения Xamarin могут создаваться средствами Visual Studio [15].

2) Высокая и около-нативная производительность создаваемых приложений [15].

Кроссплатформенное приложение, разрабатываемое с помощью фреймворка Xamarin, можно классифицировать как нативное. По проводимым тестам приложения, созданные под Xamarin 5 версии, сопоставимы с традиционными приложениями, созданными с помощью Java и Swift для Android и iOS, соответственно. На основе облачных технологий Microsoft Azure имеется возможность создавать производительные эмуляции устройств. Помимо этого, имеется возможность запускать автоматические тесты записывающие ошибки в логи, что дает возможно заблаговременно отловить ошибки и найти проблемы, ограничивающие производительность.

3) Нативный пользовательский интерфейс [15].

Использование Xamarin Shell дает возможность быстро и эффективно создавать красивые и производительные пользовательские интерфейсы, которые будут отображаться с нативной стилизацией на каждой используемой платформе. Однако имеется возможность использо-

вать отдельный рендер интерфейса для каждой платформы, что значительно повышает гибкость процесса написания приложения.

4) Совместимость с оборудованием [15].

Благодаря специальным запросам прослойкам между исходным кодом C# и конечной операционной системой обеспечивается полная совместимость всех компонентов приложения с устройством. Помимо стандартных реализаций Xamarin поддерживает множество дополнительных плагинов, которые распространяются через менеджера пакетов NuGet и корректно встраиваются даже в уже завершенное приложение, тем самым обеспечивая возможность дополнить его функциональность.

5) Технологии с открытым исходным кодом [15].

После приобретения компанией Microsoft, Xamarin стал полностью бесплатным инструментом, который распространяется по лицензии Microsoft Visual Studio, тем самым давая полную свободу разработчикам ПО.

6) Простая поддержка уже готовых решений [15].

Из-за того, что Xamarin имеет единую кодовую базу, то для внесения изменений в приложение необходимо только пересобрать исходное решение для каждой платформы. Однако учитывая, что имеется возможность вносить корректировки для каждой платформы отдельно, то желательно следить за разработкой с помощью системы контроля версий. Таким образом для команд разработчиков всегда будет понятно, для какой платформы и какие конкретно изменения были произведены.

7) Готовый инструментарий для разработки приложений [15].

Так как Xamarin распространяется через Microsoft Visual Studio, то и он является основной средой разработки. Помимо наличия самой IDE и упомянутой ранее облачной эмуляции, и хранения решения, с помощью Microsoft Azure, в инструментарий входят также аналитические инструменты.

8) Инструмент автодополнения, корректировки и отслеживания ошибок [15].

Главным преимуществом использования Microsoft Visual Studio является IntelliSense – инструмент автодополнения, корректировки и отслеживания ошибок до начала процесса компиляции. Он позволяет производить отладку приложения с отслеживанием всех параметров, функций и методов, вызываемых в определённый шаг программы.

Недостатки использования инструмента Xamarin:

1) Задержки обновления дополнительных компонентов [20-22].

Так как зачастую необходимо использовать сторонние плагины, то возникает ситуация, когда версия проекта обновилась на новую версию, а используемые в проекте плагины ее не поддерживают или работают со сбоями. В такой ситуации разработка проекта может замедлиться, вплоть до момента выпуска свежего обновления разработчиками плагина.

2) Ограниченный доступ к библиотекам с исходным кодом [20-22].

При разработке приложений с использованием нативных инструментов появляется возможность использовать все доступные технологии с исходным кодом. В случае с Xamarin остается довольствоваться только предоставляемыми компонентами, а также некоторыми реализациями функций .NET. Однако ничего не мешает написать собственные реализации тех или иных компонентов на C#. Либо в случае необходимости обратится к магазину расширений, которые включают в себя тысячи компонентов для написания логики, элементов пользовательского интерфейса, сниппеты, миксины и валидаторы.

3) Ограниченная функциональность внутренней среды платформы [20-22].

Несмотря на бурный рост Xamarin при поддержки такого гиганта как Microsoft, очевидно, что система не может предложить разработчикам весь спектр той функциональности, что могут предложить нативные инструменты. Также это относится и к общему числу разработчиков. На сегодняшний день может быть затруднительно найти опытного специалиста, имеющего

богатый опыт разработки на Xamarin. По статистическим данным, количество разработчиков на Xamarin составляет не более восьми процентов от общего числа. Однако следует отметить тот факт, что с 2016 года сообщество выросло более чем на 300%, что описывает данный инструмент как особо перспективный, а при поддержке Microsoft он будет активно развиваться в будущем.

4) Больший размер приложений [20-22].

Одной из основных проблем при компиляции приложения может быть то, что даже приложения с небольшой функциональностью могут иметь большой объем в связи с используемыми пакетами Mono, которые служат для работы приложения на конечной системе, а даже самые простые приложения по типу «Привет, мир!» могут иметь объем больше 20 Мб. Таким образом, если имеется такая необходимость, то разработчикам необходимо тратить дополнительные ресурсы для оптимизации приложения.

Обсуждение результатов. Алгоритм шифрования данных. Rijndael – это итерационный блочный симметричный шифр с архитектурой «Квадрат». Шифр имеет различную длину блоков и различные длины ключей. Длина ключа и длина блока могут быть равны: 128, 192 или 256 битам независимо друг от друга. Выявим основные преимущества алгоритма Rijndael [26].

Так как Rijndael блочный шифр, то, как и любому блочному шифру, ему соответствуют следующие принципы:

Рассеивание (diffusion) – т.е. изменение любого знака открытого текста или ключа влияет на большое число знаков шифротекста, что скрывает статистические свойства открытого текста;

Перемешивание (confusion) – использование преобразований, затрудняющих получение статистических зависимостей между шифротекстом и открытым текстом.

Rijndael не подвержен многим видам криптоаналитических атак, таких как дифференциальный и линейный криптоанализ, Square-атака, метод интерполяции и др.

Исследования, проведённые различными сторонами, показали высокое быстродействие Rijndael на различных платформах. Ценным свойством этого шифра является его байт-ориентированная структура, что обещает хорошие перспективы при его реализации в будущих процессорах и специальных схемах [27].

Некоторым недостатком можно считать то, что режим обратного расшифрования отличается от режима зашифрования порядком следования функций, и сами эти функции отличаются своими параметрами от применяемых в режиме зашифрования. Данный факт сказывается на эффективности аппаратной реализации шифра. Однако Rijndael имеет более цельную структуру, за один раунд в нём преобразуются все биты входного блока, в отличие от шифров Фейстеля, где за один раунд изменяется, как правило, лишь половина бит-входного блока. По этой причине Rijndael может позволить себе меньшее число раундов преобразования, что можно рассматривать, как некоторую компенсацию за потери при реализации режима обратного шифрования. Гибкость, заложенная в архитектуре Rijndael, позволяет варьировать не только длину ключа, что используется в стандарте AES, но и размер блока преобразуемых данных, что хотя и не нашло применения в стандарте, автором рассматривается, как вполне нормальное расширение этого шифра.

Таким образом, алгоритм Rijndael является на настоящий момент одним из наиболее надёжных и криптостойких алгоритмов [28].

Программное обеспечение. Для разработки кроссплатформенного мобильного приложения будет использоваться интегрированная среда разработки Microsoft Visual Studio, а конкретно Xamarin – фреймворк для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C# [6-8].

Проект на Xamarin.Forms представляет собой набор элементов, которые создаются средой разработки по умолчанию и участвуют в процессе компиляции и обеспечивают кроссплат-

форменность кода. Основой приложения служат страницы разметки с расширением .xaml, которые отвечают за графический интерфейс той или иной страницы приложения, и файлы логики страницы, названия которых соответствуют страницам, за которые они отвечают, но с расширением .cs.

Код файлов логики выполняется на языке программирования C#, в то время как файлы, отвечающие за графический интерфейс, редактируются с использованием расширенного языка разметки XAML.

Основные файлы, отвечающие за работу приложения, представлены в обозревателе решений, который показан на рис. 6 и включают в себя:

1) App.xaml – инициализация приложения.
2) AppShell.xaml – создание навигационного меню.
3) Storage.xaml – страница, связанная с сервисом облачного хранения файлов.
4) Encryption.xaml – страница для проведения операций шифрования/дешифрования данных.

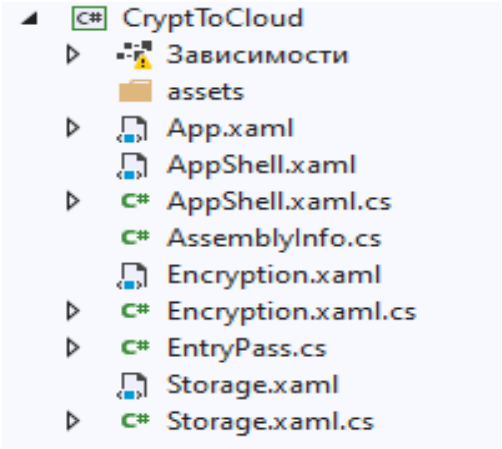

Рис. 6. Обозреватель решений
Fig. 6. Solution Explorer

App.xaml. Все инструкции, заложенные в этот файл, исполняются при запуске приложения, таким образом именно он ответственен за его инициализацию. Файл разметки не содержит кода, а в файле логики содержится единственная команда по инициализации навигационной оболочки AppShell.

AppShell.xaml. Этот файл представляет собой встроенный в среду разработки инициализатор навигационной оболочки. Оболочка Xamarin.Forms упрощает разработку мобильных приложений, предоставляя основные возможности, которые необходимы для большинства мобильных приложений, такие как:

– единое место для описания визуальной структуры приложения;
– единый пользовательский интерфейс навигации;
– схема навигации на основе URI, которая позволяет переходить на любую страницу в приложении;
– обработчик интегрированного поиска.

Кроме того, приложения оболочки получают более высокую скорость отрисовки и снижение потребления памяти [6-8].

Приложения оболочки Xamarin.Forms определяют визуальную иерархию приложения, описанную в классе, подклассом которого является класс Shell. Этот класс может состоять из трех основных объектов иерархии:

1) FlyoutItem или TabBar. Объект FlyoutItem представляет один или несколько элементов всплывающего меню. Он требуется, если шаблон навигации требует использования всплывающего меню. Объект TabBar представляет нижнюю панель вкладок. Он требуется, если

навигация в приложении начинается с нижней панели вкладок (когда всплывающее меню не требуется).

2) Tab представляет сгруппированное содержимое с навигацией по нижним вкладкам.

3) Объект ShellContent, представляющий объекты ContentPage для каждой вкладки.

Эти объекты не представляют собой какие–либо элементы пользовательского интерфейса, они служат лишь для организации визуальной структуры приложения. На основе этих объектов оболочка создает пользовательский интерфейс навигации по содержимому [6-8].

В этом проекте используется элемент Tab и основной контент приложения располагается в двух страницах, переключаться между которыми можно с помощью панели, находящийся в нижней части экрана.

Файл AppShell.xaml содержит в себе стилизацию оболочки в целом (цветовые схемы текста, фона, выделенных и не выделенных элементов):

```
BackgroundColor="#118AB2"
ForegroundColor="Black"
TitleColor="#06D6A0"
DisabledColor="#B4FFFFFF"
UnselectedColor="#95FFFFFF"
Shell.NavBarHasShadow="true"
Shell.TabBarTitleColor="#06D6A0"
```

Также в него входит сама разметка навигации:

```
<TabBar>
<Tab Title="Хранилище"
Icon="storage_icon.png">
<ShellContent>
<app3:Site />
</ShellContent>
</Tab>
<Tab Title="Шифрование"
Icon="crypt_icon.png">
<ShellContent>
<app3:Encryption />
</ShellContent>
</Tab>
</TabBar>
```

Таким образом тег TabBar включает в себя 2 вкладки Tab, содержащие информацию о заголовке и иконке вкладки, а также имя страницы содержимого для привязывания вкладки к ней.

Файл логики страницы содержит в себе только команду для самой инициализации страницы.

Storage.xaml. Данный файл представляет собой страницу доступа к веб–сервису облачного хранилища. Он работает путем открытия в этой вкладке страницы с помощью встроенного в систему (Android или iOS) браузера. Достигается это благодаря имеющимся в библиотеке Xamarin.Forms тегу WebView.

```
<WebView x:Name="webView" Navigating="WebView_OnNavigating"
Source="https://serikov.tk/cloudshot/" VerticalOptions="FillAndExpand" />
```

Encryption.xaml. В данном файле содержится наибольшая доля кода, как с точки зрения графического пользовательского интерфейса, так и с точки зрения программной логики. Страница включает в себя функциональность шифрования и дешифрования данных.

Графический интерфейс представляет собой поле для ввода ключа шифрования, а также кнопки для шифрования и дешифрования выбранных данных.

Поле ввода является специально стилизованным и прописывается отдельно для каждой используемой платформы через класс EntryPassRenderer, который наследуется от базового EntryRenderer, позже он вызывается в основной интерфейс. Также для этого поля назначен обработчик события (TextChanged), который обновляет переменную, хранящую в себе текст, введенный в поле.

```
<local:EntryPass x:Name="PasswordEntry" TextChanged="PassChanged"
Placeholder="Введите ключ шифрования" HorizontalOptions="Center" WidthRequest="350" Hori-
zontalTextAlignment="Center" PlaceholderColor="#073B4C" IsPassword="True"
></local:EntryPass>
```

Кнопки реализуются с помощью тега Button, в свойствах которых указываются стилевые особенности, имя элемента (x:Name), а также имя обработчика событий (Clicked) при нажатии пользователем этой кнопки.

Стилевые особенности для элемента типа Button были установлены глобально для всех элементов такого типа на странице посредством тега ResourceDictionary и включенного в него тега Style.

```
<ResourceDictionary>
<Style x:Key="buttonStyle" TargetType="Button">
<Setter Property="TextColor" Value="#06D6A0" />
<Setter Property="BackgroundColor" Value="#073B4C" />
<Setter Property="FontSize" Value="Large" />
<Setter Property="CornerRadius" Value="30"></Setter>
<Setter Property="WidthRequest" Value="150"></Setter>
<Setter Property="FontAttributes" Value="Bold"></Setter>
</Style>
</ResourceDictionary>
```

Файл логики страницы Encryption.xaml.cs включает в себя обработчики событий при взаимодействии с графическим интерфейсом, метод шифрования данных, а также переменные, необходимые для корректного функционирования кода.

При нажатии на кнопку (Шифрование/Дешифрование) происходит выбор файла, с которым будут проводится операции. Выбор файла осуществляется посредством встроенного в используемой системе инструмента и происходит с помощью дополнительно подключаемой кроссплатформенной библиотеки Plugin.FilePicker.

При выборе файла, полный путь к файлу передается в переменную, и после вызывается метод Encrypt/Decrypt. Методы Encrypt/Decrypt представляют собой реализацию алгоритма Rijndael с помощью стандартных криптосервисов .NET Core.

.NET Core включает набор криптографических сервисов, расширяющих аналогичные сервисы Windows и других поддерживаемых платформ через CryptoAPI. Пространство имен System.Security.Cryptography открывает программный доступ к самым разнообразным криптографическим сервисам, с помощью которых приложения могут шифровать и дешифровать данные, обеспечивать их целостность, а также обрабатывать цифровые подписи и сертификаты [6-8].

Вывод. В ходе работы был представлен алгоритм применения кроссплатформенного мобильного приложения для защиты информации в «облачном хранилище» ИС, а также обосновано его применение для хранения резервных копий данных. С учетом распространения смартфонов предлагается их использование в качестве терминальных устройств для выгрузки резервных копий файлов в «облачное хранилище» с применением кроссплатформенного мобильного приложения, которое бы могло компилироваться и эффективно работать под управ-

лением операционных систем iOS и Android, выполнять шифрование файлов и выгружать их в «облачное хранилище» с возможностью последующей дешифровки. Проведен анализ существующих подходов и фреймворков разработки кроссплатформенных мобильных приложений и было принято решение создавать приложение на основе фреймворка Xamarin [14-16].

Предлагаемое приложение связано с виртуальным выделенным сервером, которое выступает в роли «облачного хранилища». Поддерживается предварительное шифрование и последующие дешифрование по стандарту AES (алгоритм шифрования Rijndael) [23].

Библиографический список:

1. Aws.Amazon. Что такое облачное хранилище файлов? [Электронный ресурс]. URL: https://aws.amazon.com/ru/what-is-cloud-file-storage/ (дата обращения: 25.06.2022).
2. Blesson Varghese, Rajkumar Buyya Next Generation Cloud Computing: New Trends and Research Directions, Future Generation Computer Systems, 2017.
3. EMC Corporation. Резервное копирование и архивирование: Учебное пособие. 2012. – 56 с.
4. Flutter. официальный сайт разработчиков Flutter. [Электронный ресурс]. URL: https://flutter.dev/ (дата обращения: 25.06.2022).
5. iXBT. Виртуализация: новый подход к построению IT-инфраструктуры [Электронный ресурс]. URL: https://www.ixbt.com/cm/virtualization.shtml (дата обращения: 25.06.2022).
6. Jiyi WU, Lingdi PING Cloud Storage as the Infrastructure of Cloud Computing, International Conference on Intelligent Computing and Cognitive Informatics, 2010.
7. Metanit.com. Руководство по программированию для Xamarin Forms. [Электронный ресурс]. https://metanit.com/sharp/xamarin/ (дата обращения: 25.06.2022).
8. Microsoft Docs. Выполнение дедупликации данных [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/windows-server/storage/data-deduplication/run (дата обращения: 25.06.2022).
9. Microsoft Docs. Документация по Xamarin. [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/xamarin/ (дата обращения: 25.06.2022).
10. PhoneGap. официальный сайт разработчиков PhoneGap. [Электронный ресурс]. URL: http://www.build.phonegap.com (дата обращения: 25.06.2022).
11. React Native. официальная страница React Native на сайте GitHub. [Электронный ресурс]. URL: https://facebook.github.io/react-native/ (дата обращения: 25.06.2022).
12. Statista: Доходы от мобильных приложений по всему миру. [Электронный ресурс]. URL: https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast (дата обращения: 25.06.2022).
13. Алиев А. А., Самедов Р. Б. Алгоритм создания полного резервного копирования в облачных вычислениях // Вестник Бакинского Университета, 2013 - 306 с.
14. Андреевский И. Л. Технологии облачных вычислений: Учебное пособие. – СПб. : Изд-во СПбГУ, 2018. – 80 с.
15. В. В. Бастрикина Обзор фреймворка Xamarin // Моделирование и анализ информационных систем – Красноярск. : Изд-во Сибирский государственный университет науки и технологий, 2015 – 579 с.
16. В.Г. Казаков, С.А. Федосин Технологии и алгоритмы резервного копирования. – М. : Изд-во МГУ, 2016 – 40 с.
17. Гребнев Е. Облачные сервисы: взгляд из России, Издательство: Cnews, 2011, с. 282.
18. Клементьев И. П. Устинов В. А. Введение в Облачные вычисления, Изд-во : УГУ, 2009 – 223 с.
19. Ковалева О.В. Облачные хранилища данных, их особенности и модели обслуживания // Информационные технологии. Эволюционные процессы – Королев : Изд-во МГОТУ, 2018. – 130 с.
20. Леонов В. GoogleDocs, WindowsLive и другие облачные технологии, Изд-во: Эксмо-Пресс, 2012 – 304 с.
21. Резервное копирование (Backup) // Администрирование серверов и техническая поддержка сайтов. [Электронный ресурс]. Режим доступа: https://itfb.com.ua/backup/ (дата обращения: 25.06.2022).
22. Савин И.В. Инкрементальное резервное копирование. Преимущества и недостатки // Современные инновации – Иваново: Изд-во Олимп, 2018 – 80 с.
23. Свентицкий П. Инструменты кроссплатформенной разработки мобильных приложений // Инновации в науке: сб. ст. по матер. XL междунар. науч.-практ. конф. – Новосибирск. : Изд-во СибАК, 2014 – 402 с.
24. Таранин С. М. Резервное копирование с хранением в базе данных // Моделирование и анализ информационных систем – Ярославль. : Изд-во Ярославский государственный университет, 2016 – 507 с.
25. Хабр. Архитектура хранилищ данных: традиционная и облачная [Электронный ресурс]. URL: https://habr.com/ru/post/441538/ (дата обращения: 25.06.2022).
26. Хабр. Обзор IT-рынка облачных решений для бизнеса [Электронный ресурс]. URL: https://habr.com/ru/post/417193/ (дата обращения: 25.06.2022).
27. Хабр. Описание основ криптопреобразования AES [Электронный ресурс]. URL: https://habr.com/ru/post/497672/ (дата обращения: 25.06.2022).
28. Шарапов Р.В. Аппаратные средства хранения больших объёмов данных // Инженерный вестник Дона. – Р-н-Д. : Изд-во Южный федеральный университет, 2012 – 304 с.

References:

1. Aws.Amazon. What is cloud file storage? [electronic resource]. URL: https://aws.amazon.com/ru/what-is-cloud-file-storage / (accessed: 06/25/2022).

2. Blesson Varghese, Rajkumar Buyya Next Generation Cloud Computing: New Trends and Research Directions, Future Generation Computer Systems, 2017.
3. EMC Corporation. Backup and Archiving: A tutorial. 2012: 56.
4. Flutter. official website of Flutter developers. [electronic resource]. URL: https://flutter.dev / (accessed: 06/25/2022).
5. iXBT. Virtualization: a new approach to building IT infrastructure [Electronic resource]. URL: https://www.ixbt.com/cm/virtualization.shtml (accessed: 06/25/2022).
6. Jiyi WU, Lingdi PING Cloud Storage as the Infrastructure of Cloud Computing, International Conference on Intelligent Computing and Cognitive Informatics, 2010.
7. Metanit.com. Programming Guide for Xamarin Forms. [Electronic resource]. https://metanit.com/sharp/xamarin / (accessed: 06/25/2022).
8. Microsoft Docs. Performing data deduplication [Electronic resource]. URL: https://docs.microsoft.com/ru-ru/windows-server/storage/data-deduplication/run (accessed: 06/25/2022).
9. Microsoft Docs. Xamarin documentation. [Electronic resource]. URL: https://docs.microsoft.com/ru-ru/xamarin / (accessed: 06/25/2022).
10. PhoneGap. the official website of PhoneGap developers. [Electronic resource]. URL: http://www.build.phonegap.com (accessed: 06/25/2022).
11. React Native. The official React Native page on GitHub. [Electronic resource]. URL: https://facebook.github.io/react-native / (accessed: 06/25/2022).
12. Statista: Mobile app revenue worldwide. [Electronic resource]. URL: https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast (accessed: 06/25/2022).
13. Aliev A. A., Samedov R. B. Algorithm for creating a full backup in cloud computing [Vestnik Bakinskogo Universiteta] *Bulletin of Baku University*, 2013: 306.
14. Andreevsky I. L. Cloud Computing Technologies: A textbook. St. Petersburg : Publishing House of St. Petersburg State University, 2018: 80. (In Russ)
15. V. V. Bastrikina Overview of the Xamarin framework. [Modelirovaniye i analiz informatsionnykh sistem] *Modeling and analysis of information systems.* Krasnoyarsk. : Publishing house of the Siberian State University of Science and Technology, 2015:579. (In Russ)
16. V.G. Kazakov, S.A. Fedosin Technologies and algorithms of backup. M. : Publishing House of Moscow State University, 2016: 40. (In Russ)
17. Grebnev E. Cloud services: a view from Russia, Publishing House: Cnews, 2011: 282. (In Russ)
18. Klementyev I. P. Ustinov V. A. Introduction to Cloud computing, Publishing House : UGU, 2009: 223. (In Russ)
19. Kovaleva O.V. Cloud data warehouses, their features and service models. [Informatsionnyye tekhnologii. Evolyutsionnyye protsessy] *Information technologies. Evolutionary processes/* Korolev : MGOTU Publishing House, 2018:– 130. (In Russ)
20. Leonov V. GoogleDocs, Windows Live and other cloud technologies, Publishing House: Eksmo-Press, 2012: 304. (In Russ)
21. Backup. Server administration and technical support of sites. [Electronic resource]. Access mode: https://itfb.com.ua/backup / (accessed: 06/25/2022).
22. Savin I.V. Incremental backup. Advantages and disadvantages. [Sovremennyye innovatsii*] Modern innovations* – Ivanovo. : Olympus Publishing House, 2018: 80. (In Russ)
23. Sventitsky P. Tools for cross-platform development of mobile applications. Innovations in science: collection of articles on Mater. XL International Scientific and Practical Conference – Novosibirsk. : SibAK Publishing House, 2014: 402. (In Russ)
24. Taranin S. M. Backup with storage in a database [Modelirovaniye i analiz informatsionnykh sistem] *Modeling and analysis of information systems* – Yaroslavl. : Publishing house of Yaroslavl State University, 2016: 507. (In Russ)
25. Habr. Data storage architecture: traditional and cloud [Electronic resource]. URL: https://habr.com/ru/post/441538 / (accessed: 06/25/2022).
26. Habr. Overview of the IT market of cloud solutions for business [Electronic resource]. URL: https://habr.com/ru/post/417193 / (accessed: 06/25/2022).
27. Habr. Description of the basics of AES crypto conversion [Electronic resource]. URL: https://habr.com/ru/post/497672 / (accessed: 06/25/2022).
28. Sharapov R.V. Hardware for storing large amounts of data [Inzhenernyy vestnik Dona] *Engineering Bulletin of the Don*. – R-n-D. : Publishing House of the Southern Federal University, 2012: 304. (In Russ)

Сведения об авторе:
Газизов Андрей Равильевич, кандидат педагогических наук, доцент, кафедра «Вычислительные системы и информационная безопасность»; gazandre@yandex.ru

Information about author:
Andrey R. Gazizov, Cand. Sci. (Pedagogical), Assoc. Prof., Department of Computing Systems and Information Security; gazandre@yandex.ru

Конфликт интересов/Conflict of interest.
Автор заявляет об отсутствии конфликта интересов/The author declare no conflict of interest.
Поступила в редакцию/Received 08.05.2022.
Одобрена после рецензирования/ Reviced 30.05.2022.
Принята в печать/Accepted for publication 30.05.2022.