НОМЕРКИ СТРАНИЧЕК КАК СДЕЛАЕШЬ ДНЕВНИК ТО ЧТО ТЕБЕ НУЖНО. ПОМЕНЯЙ НА СВОИ, СВЕРЯЙСЯ С НАЗВАНИЯМИ, ЛИБО СКИНЬ МНЕ КОГДА СДЕЛАЕШЬ ТАКЖЕ И С ХАРАКТЕРИСТИКАМИ МЕСТА ПРАКТИКИ; ТУТ В ПРИНЦИПЕ СТРАНИЦЫ НАДО В СОДЕРЖАНИИ ПОМЕНЯТЬ В СООТВЕТСТВИИ С РЕАЛНОСТЬЮ)

Содержание

Введение 12

1 Характеристика места практики 14

2 Сведения об имеющейся вычислительной технике и программном обеспечении 15

3 Сведения о прохождении практики в качестве дублера инженерно-технического работника 16

4 Индивидуальное задание 17

4.1 Постановка задачи 17

4.2 Проектирование структуры программы 18

4.2.1 Desktop версия приложения 18

4.2.2 Мобильная версия приложения 18

4.3 Функциональное взаимодействие модулей программы 19

4.3.1 Desktop версия приложения 19

4.3.2 Мобильная версия приложения 21

4.4 Работа с программой 22

4.4.1 Desktop версия приложения 25

4.4.2 Мобильная версия приложения 30

Заключение 36

Список использованных источников 38

Приложения

А – Основные процедуры desktop приложения 39

Б – Основные процедуры Мобильного приложения 41

Б – Основные процедуры unit-тестов 45

ВВЕДЕНИЕ

Целью производственной практики по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем является закрепление и совершенствование приобретенных в процессе обучения профессиональных умений обучающихся по специальности, развитие общих и профессиональных компетенций, адаптация к конкретным условиям деятельности предприятий и организаций, освоение современных производственных процессов, обучение трудовым приемам, операциям и способам выполнения трудовых процессов.

В процессе прохождения производственной практики необходимо будет решить следующие задачи:

* приобрести практический опыт разработки алгоритма решения поставленной задачи и реализации его средствами автоматизированного проектирования, использования инструментальных средств на этапе отладки программного продукта и проведения тестирования программного модуля по определенному сценарию;
* приобрести практический опыт анализа алгоритмов, в том числе с применением инструментальных средств, разработки кода программного продукта на основе готовой спецификации на уровне модуля, осуществления рефакторинга и оптимизации программного кода;
* приобрести практический опыт разработки мобильных приложений;
* сформировать профессиональные умения и навыки;
* приобрести практические навыки в будущей профессиональной деятельности при выполнении функций дублера инженерно-технических работников;
* выполнить индивидуальное задание.

В результате прохождения производственной практики необходимо будет подтвердить уровень овладения видом деятельности «Разработка модулей программного обеспечения для компьютерных систем» и следующими профессиональными компетенциями:

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4 Выполнять тестирование программных модулей.

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода.

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.

Производственная практика проводится на базе арбитражного суда республики Адыгея.

3 СВЕДЕНИЯ О ПРОХОЖДЕНИИ ПРАКТИКИ В КАЧЕСТВЕ ДУБЛЕРА ИНЖЕНЕРНО-ТЕХНИЧЕСКОГО РАБОТНИКА

Во время прохождения практики в качестве дублера техника-программиста я выполнял следующие действия:

* ознакомление с системным и прикладным программным обеспечением на предприятии;
* работа с документацией;
* сбор и структурирование материала по теме индивидуального задания «Магазин игрушек»
* анализ алгоритмов, в том числе с применением инструментальных средств;
* разработка кода программного продукта на основе готовой спецификации на уровне модуля;
* написание программного кода, реализующего функциональность разрабатываемого приложения;
* разработка интерфейса приложения;
* отладка программного продукта с использованием спе­циализированных программных средств;
* разработка тестовых наборов и тестовых сценариев;
* осуществление рефакторинга и оптимизации программного кода;
* тестирование программного продукта;
* разработка технической документации к программному продукту;
* обслуживание оргтехники;
* установка и настройка необходимого программного обеспечения;
* разработка кода по заданию руководителя практики.

Выполненная работа соответствует профессиональным компетенциям, предусмотренным программой производственной практики ПМ.01 специальности 09.02.07 Информационные системы и программирование.

4 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

4.1 Постановка задачи

В период практики необходимо написать desktop-приложение реализующее функцию просмотра информации из магазина игрушек «BimBom», а также версию данного приложения для мобильных платформ. Программа должна содержать текстовую и графическую информацию, иметь удобный и понятный пользователю графический интерфейс, быть проста в использовании.

Программа должна предлагать следующие возможности:

-просмотр информации и цены на товар

-Просмотр своего профиля

-Редактирование и добавление нового товара

-вывод информации и изображений в текстовые и графические поля.

4.2 Проектирование структуры программы

4.2.1 Desktop версия приложения

Разработанное приложение имеет следующие окна:

* окно авторизации;
* главное окно, которое содержит области:
* информацию о имеющимся товаре
* Кнопки «Редактирования» и «Добавления» Товара

4.2.2 Мобильная версия приложения

Разработанное приложение имеет следующие окна:

* окно заставки;
* окно авторизации/регистрации;
* окно с профилем;
* окно с информацией о товаре;

4.3 Функциональное взаимодействие модулей программы

4.3.1 Desktop версия приложения

На основе разработанного технического проекта был написан программный продукт «Магазин игрушек». Для создания приложения была использована среда программирования Blend for Visual Studio 2019 и язык программирования С#. В программе используются следующие языковые модули:

* System− содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Windows− содержит типы, используемые в приложениях
* System.IO −cодержит типы, поддерживающие ввод и вывод, включая возможности чтения и записи данных в потоках как синхронно, так и асинхронно, реализации пользовательского ведения журнала и обработки входящих и исходящих потоков данных в последовательных портах;
* System.Data− содержит типы для доступа к данным из различных источников и для управления этими данными. Пространство имен верхнего уровня и несколько дочерних пространств имен образуют архитектуру ADO.NET и поставщиков данных ADO.NET [9].
* System.Threading− содержат типы, обеспечивающие возможности многопотокового программирования. Дочернее пространство имен предоставляет типы, которые упрощают задачу написания параллельного и асинхронного кода;
* System.Collections.Generic – работа с обобщёнными коллекциями (List<string>) [5].

В программе были созданы следующие пользовательские модули:

* MainWindow.xaml – окно авторизации;
* EditWindow.xaml – Окно с редактированием товара;
* BimBomWindow.xaml – Главное меню приложения со списком товаров;
* AddToysWindow.xaml – Добавление новой игрушки;

В модуле MainWindow.xaml.cs описаны и используются следующие процедуры:

private void Image\_MouseDown(object sender, MouseButtonEventArgs e) – Показать/Скрыть пароль.

В модуле EditWindow.xaml.cs описаны и используются следующие процедуры:

public EditWindow(Toy toy) – Добавление имеющихся данных о товаре;

private void DeleteBut\_Click(object sender, RoutedEventArgs e) – Удаление товара;

private void saveBut\_Click(object sender, RoutedEventArgs e) – Сохранение информации в базе данных о товаре;

private void imageBut\_Click(object sender, RoutedEventArgs e) – Добавление новой картинки

В модуле BimBomWindow.xaml.cs описаны и используются следующие процедуры:

private void editBut\_Click(object sender, RoutedEventArgs e) – Открывает окно с редактированием товара;

В модуле AddToysWindow.xaml.cs описаны и используются следующие процедуры:

private void saveBut\_Click(object sender, RoutedEventArgs e) – Сохранение информации в базе данных о товаре;

private void imageBut\_Click(object sender, RoutedEventArgs e) – Добавление картинки

private void priceText\_KeyDown(object sender, KeyEventArgs e) – Запрет ввода букв в textbox;

В модуле MainWindowViewModel.cs описаны и используются следующие процедуры:

public MainWindowViewModel() – Добавление активного пользователя в базу данных;

private void MainButtonClick() – Авторизация;

Код данных процедур представлен в приложении А.

4.3.2 Мобильная версия приложения

На основе разработанного технического проекта была написана мобильная версия программного продукта «ИС Магазин игрушек». Для создания приложения была использована среда программирования Android Studio и язык программирования Kotlin. В программе используются следующие языковые модули:

− android.content.Context – это базовый абстрактный класс, реализация которого обеспечивается системой Android. Этот класс имеет методы для доступа к специфичным для конкретного приложения ресурсам и классам и служит для выполнения операций на уровне приложения, таких, как запуск активностей, отправка широковещательных сообщений, получение намерений и прочее;

− android.graphics.Canvas – предоставляет методы для рисования, которые отображают графические примитивы на исходном растровом изображении;

− android.graphics.Color – содержит несколько констант и методов для конвертации и извлечения цветового компонента;

− android.graphics.Paint – содержит стили, цвета и другую графическую информацию для рисования графических объектов;

− android.view.View – компонент(виджет), который рисуется на экране;

− java.util.ArrayList – класс, позволяющий создавать списки;

− android.app.AlertDialog – это уведомление, которое появляется, когда определенное приложение использует систему Android для размещения важной информации для пользователя;

− android.content.Intent – класс, объекты которого используются для взаимодейтвия между отдельными частями Android приложения, либо между различными приложениями системы;

− android.os.Bundle – необходим для временного хранения данных в процессе выполнения;

− android.view.Menu – класс, представляющий меню в приложениях;

− android.view.MenuItem – класс, отвечающий за пункты меню;

− android.widget.Toast – это небольшое, всплывающее сообщение, появляющееся внизу и экрана на непродолжительное время;

− androidx.appcompat.app.AppCompatActivity – это базовый класс для действий, которые используют функциональные возможности панели поддержки библиотеки;

− android.widget.EditText – это стандартный виджет ввода текста в приложениях Android;

− kotlin.coroutines.CoroutineContext – все построители сопрограмм, такие как launch и async, принимают необязательный параметр CoroutineContext, который можно использовать для явного указания диспетчера для новой сопрограммы и других элементов контекста.

− android.os.AsyncTask – предлагает простой и удобный механизм для перемещения трудоёмких операций в фоновый поток.

В программе были созданы следующие пользовательские модули:

* activity\_main.xml – окно авторизации;
* activity\_registration.xml – окно регистрации;
* activity\_second – Главное окно приложение с навигацией по фрагментам;
* activity\_user – Окно с вашим профилем;
* fragment\_home – Окно со списком игрушек;
* fragment\_notifications.xml – Окно с вашим профилем;
* list\_item.xml – Список товаров;

В модуле MainActivity.kt описаны и используются следующие процедуры:

fun authorization(view: android.view.View) – Авторизация

В модуле RegistrationActivity.kt описаны и используются следующие процедуры:

fun reg(view: android.view.View) – Проверка данных на корректность и регистрация.

В модуле NotificationsFragment.kt описаны и используются следующие процедуры:

override fun onCreateView – Добавление данных о текущем пользователе.

4.4Работа с приложением

4.4.1 Desktop версия приложения

При запуске программы открывается авторизация (рисунок 1).

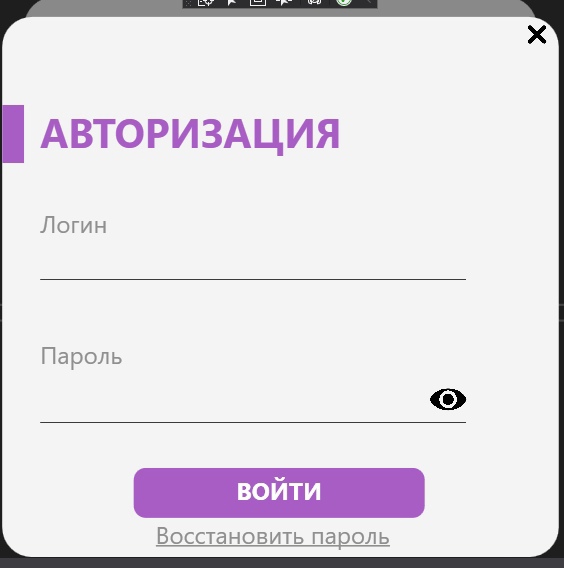


Рисунок 1 – Авторизация

В окне авторизации вы можете ввести ваш логин и пароль и войти в приложение. При неправильном вводе данных, вам высветиться соответствующее диалоговое окно (Рисунок 2). Также если вы забыли пароль, вы можете нажать на кнопку «Восстановить пароль», вам высветиться диалоговое окно с подсказкой (Рисунок 3).

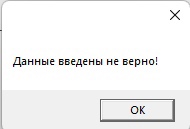


Рисунок 2. Диалоговое окно с информацией о неправильности введённых данных.

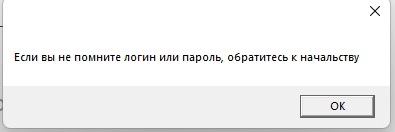


Рисунок 3. Диалоговое окно с информацией, что делать при забытом логине и пароле.

Также, вы можете просматривать введённый вами пароль (Рисунок 4, рисунок 5).



Рисунок 4. Скрытый пароль



Рисунок 5. Видимый пароль.

После успешной авторизации, у вас откроется главное меню приложение со списком имеющихся товаров.(Рисунок 6)

Главное окно имеет следующие элементы:

-Список ваших товаров;

-Кнопка «Редактировать товар»;

-Кнопка «Добавить товар»;

-Информация о активном пользователе;

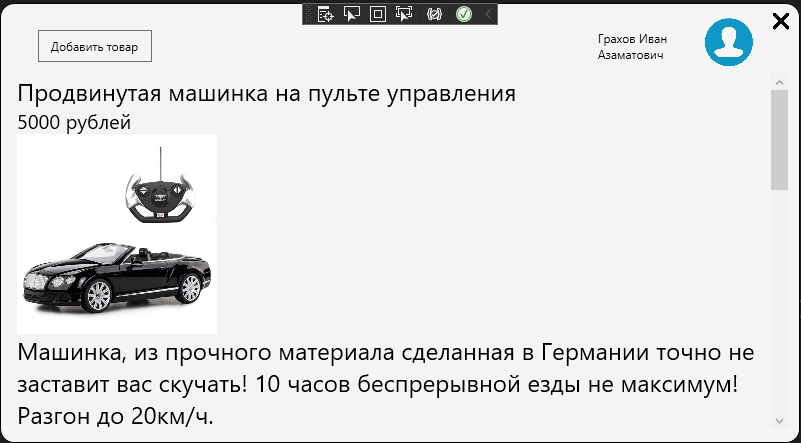


Рисунок 6. Главное окно приложения.

При нажатии на кнопку «Редактировать товар», откроется окно с информацией о выбранном товаре, которую вы можете редактировать (Рисунок 7).

При нажатии на кнопку «Добавить товар», откроется окно, в котором вы можете ввести данные о новом товаре (Рисунок 8). Далее они будут добавлены в базу данных (Рисунок 9), если вы ввели всё правильно, если вы не заполнили любой из пунктов вы увидите соответствующую ошибку (Рисунок 10).



Рисунок 7. Редактирование товара.



Рисунок 8. Создание нового товара.

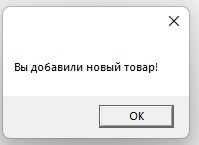


Рисунок 9. Диалоговое окно, сообщающее об успехе добавления товара.

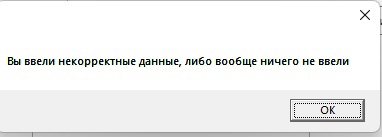


Рисунок 10. Диалоговое окно, сообщающее об ошибке.

4.4.2 Мобильная версия приложения

После запуска приложения откроется окно с авторизацией (Рисунок 11), в котором будут следующие элементы:

-Поля для ввода данных;

-Кнопка входа;

-Кнопка регистрации;

-Кнопка входа как гость;

При вводе неккоректных данных, в поле авторизации или регистрации вы увидите соответствующие всплывающие сообщения об ошибке (Рисунок 12);



Рисунок 11. Окно авторизации.

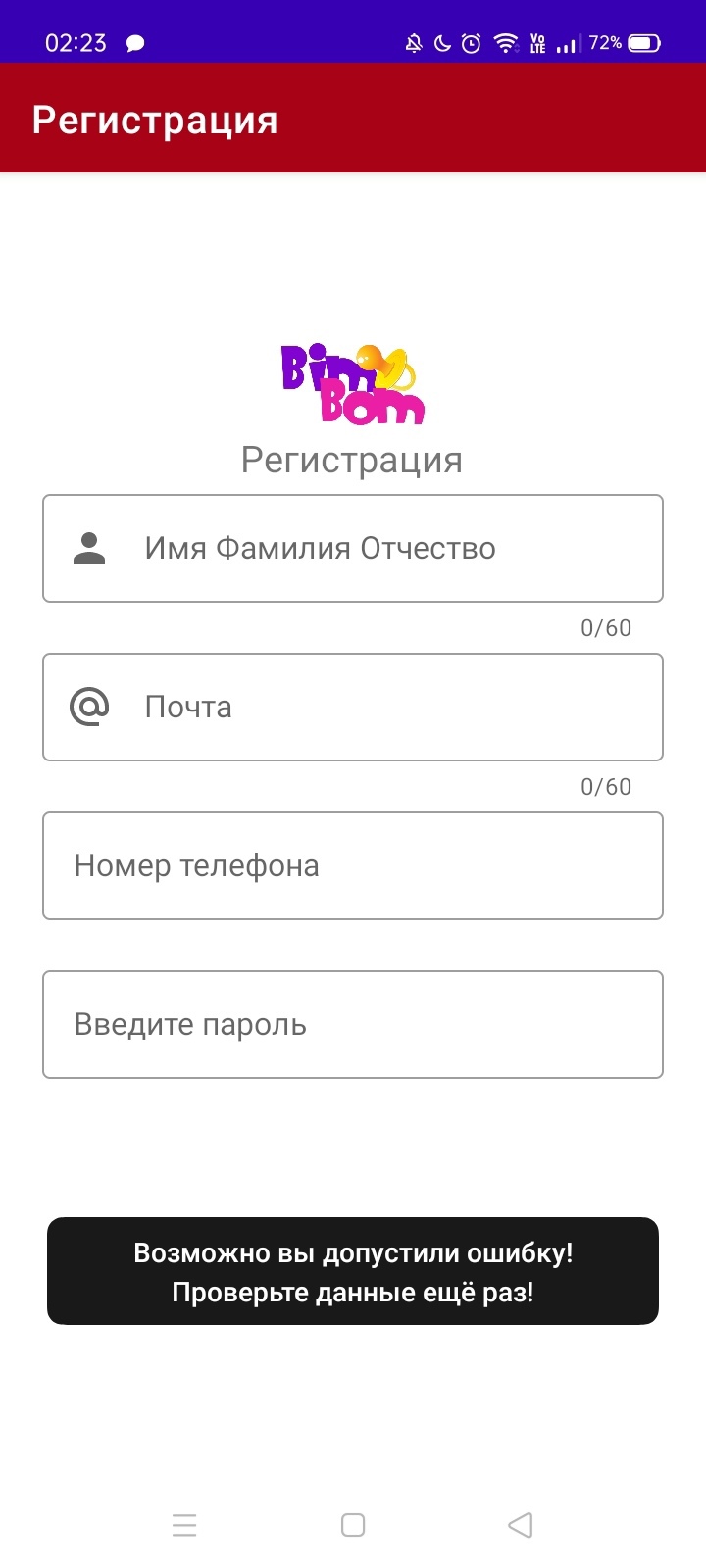


Рисунок 12. Сообщение с информацией об ошибке.

При нажатии на кнопку «Зарегестрироваться» у вас откроется окно с регистрацией, в котором вы должны правильно заполнить все данные в полях (Рисунок 13).

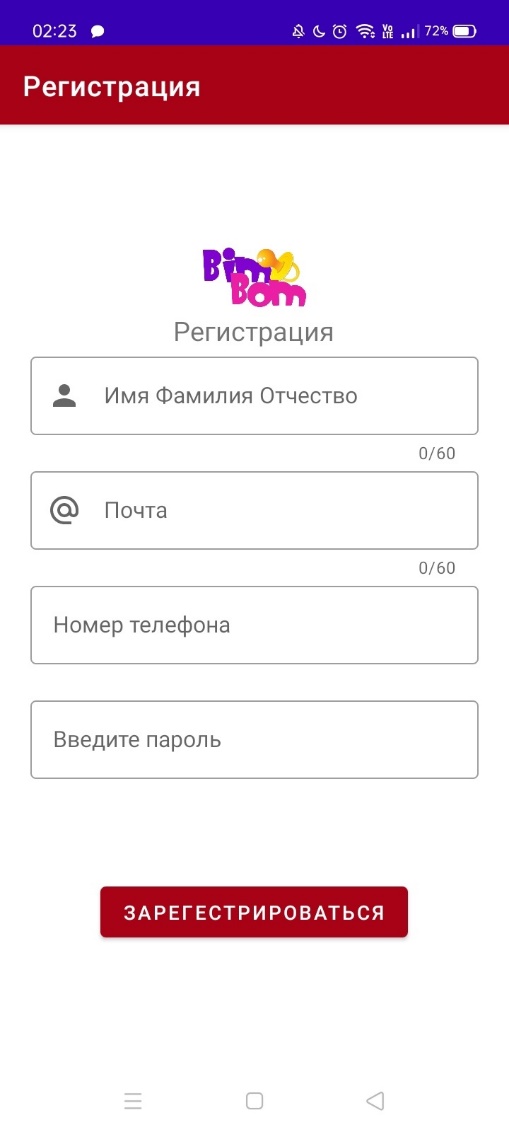


Рисунок 13. Окно с регистрацией.

После успешной авторизации/регистрации вы переходите в главное окно приложения(Рисунок 14), которое содержит следующие элементы:

-Навигационная панель снизуl

-Список товара;

Во второй вкладке, вы можете просмотреть информацию о вашем профиле (Рисунок 15).

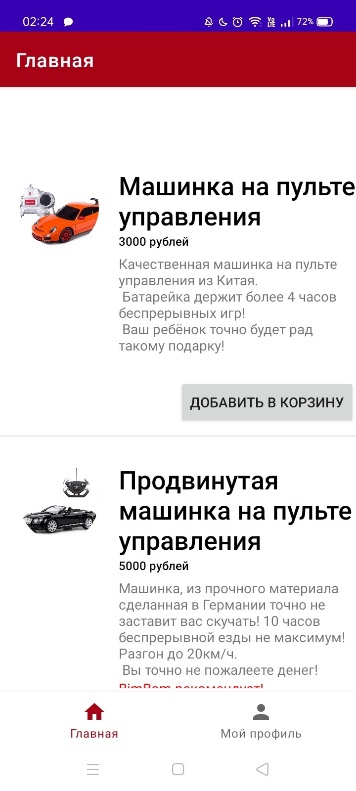


Рисунок 14. Главное окно приложения.

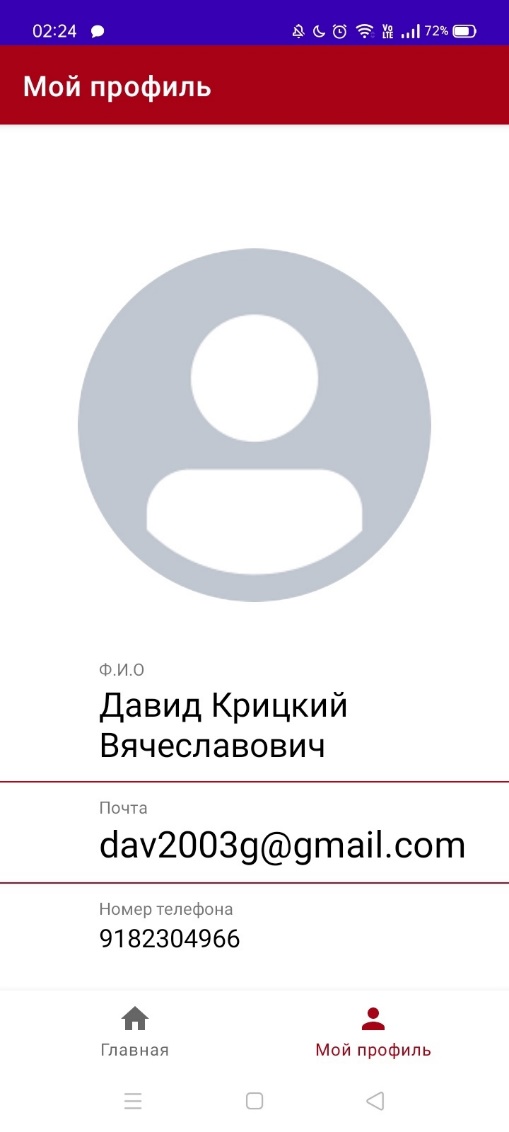


Рисунок 15. Окно с активным профилем.

ЗАКЛЮЧЕНИЕ

В результате прохождения производственной практики мною были закреплены теоретические знания, полученные при изучении ПМ.01; отработаны профессиональные компетенции по получаемой специальности; приобретены практические навыки в будущей профессиональной деятельности; выполнено индивидуальное задание.

Мною было разработано приложение «ИС магазин игрушек»

Программа имеет удобный и понятный пользователю графический интерфейс, проста в использовании и обеспечивает следующие возможности:

-Ввод данных для авторизации;

-Авторизация;

-Ввод данных для регистрации;

-Вывод информации в диалоговом окне;

-Показ/Скрытие пароля;

-Просмотр расписания после выбора: Группы, недели и дня недели;

-Просмотр своего профиля.

В ходе выполнения разработки данного приложения были:

* изучены некоторый список товаров «BimBom»;
* спроектирована структура приложения и программного кода;
* разработаны справочные материалы;
* выбраны средства реализации приложения;
* разработан программный код приложения;

Основными достоинствами программы являются:

* программа проста в установке, настройке и эксплуатации;
* данный программный продукт предназначен большого круга людей;
* наличие справочной системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://bim-bom.toys/
2. https://metanit.com/

Приложение А

Основные процедуры desktop – приложения.

private void Image\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (but==0)

{

pwdTextBox.Visibility = Visibility.Visible;

PasswordBox.Visibility = Visibility.Hidden;

but++;

return;

}

else

if (but==1)

{

pwdTextBox.Visibility = Visibility.Hidden;

PasswordBox.Visibility = Visibility.Visible;

but--;

}

}

private void imageBut\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog avatar = new OpenFileDialog();

avatar.Title = "Выберете картинку";

avatar.Filter = "Image Files (\*.bmp;\*.png;(.jpg)|\*.bmp;\*.png;\*.jpg";

if (avatar.ShowDialog() == true)

{

BitmapImage image = new BitmapImage(new Uri(avatar.FileName));

Image.Source = image;

} private void TextBox\_KeyDown(object sender, KeyEventArgs e)

{

if ((e.Key >= Key.A && e.Key <= Key.Z))

{

e.Handled = true;

}

}

private void saveBut\_Click(object sender, RoutedEventArgs e)

{

Toy save = null;

using (DataContext context = new DataContext())

{

save = context.Toys.Where(x => x.id == toy.id).FirstOrDefault();

if (save == null || string.IsNullOrEmpty(nameText.Text) || string.IsNullOrEmpty(priceText.Text) || string.IsNullOrEmpty(descText.Text))

{

MessageBox.Show("Вы ввели некорректные данные, либо вообще ничего не ввели");

}

else {

save.Name = nameText.Text;

save.Price = priceText.Text;

save.Description = descText.Text;

try

{

using (FileStream fileStream = new FileStream(Directory.GetCurrentDirectory() + "\\Image\\" + save.id + ".png", FileMode.Create))

{

BitmapFrame frame = BitmapFrame.Create((BitmapSource)Image.Source);

BitmapEncoder encoder = new PngBitmapEncoder();

encoder.Frames.Add(frame);

encoder.Save(fileStream);

}

}

catch { Exception n; }

save.ImagePath = "\\Image\\" + save.id + ".png";

context.SaveChanges();

BimBomWindow bimBomWindow = new BimBomWindow();

bimBomWindow.Show();

this.Close();

}

}

}

private void DeleteBut\_Click(object sender, RoutedEventArgs e)

{

using (DataContext context = new DataContext())

{

if (toy != null)

{

Toy game = context.Toys.Where(x => x.id == toy.id).FirstOrDefault();

context.Toys.Remove(game);

context.SaveChanges();

this.Close();

BimBomWindow bimBomWindow = new BimBomWindow();

bimBomWindow.Show();

MessageBox.Show("Данные успешно удалены!");

}

}

} private void editBut\_Click(object sender, RoutedEventArgs e)

{

var item = (sender as FrameworkElement).DataContext;

int index = ListView.Items.IndexOf(item);

EditWindow editWindow = new EditWindow(Toys[index]);

this.Close();

editWindow.Show();

}

public MainWindowViewModel()

{

User authorization = null;

using (Data.DataContext context = new Data.DataContext())

{

foreach (User user in Users)

{

string hashPass = user.Password;

authorization = context.Users.Where(x => x.Name == user.Name).FirstOrDefault();

if (authorization == null)

{

User imya = user;

context.Users.Add(imya);

context.SaveChanges();

}

}

}

LoginButtonCommand = new RelayCommand(o => MainButtonClick());

MinimizeWindowCommand = new RelayCommand(o => MinimizeWindowClick());

CloseWindowCommand = new RelayCommand(o => CloseWindowClick());

}

//Авторизация при нажатии на кнопку

private void MainButtonClick()

{

foreach (User user in Users)

{

if (user.Name == \_login && HashCode.GetHashCode(\_password) == user.Password)

{

CurrentUser.Name = user.UserName;

View.BimBomWindow controlwindow = new View.BimBomWindow();

Application.Current.MainWindow.Hide();

controlwindow.ShowDialog();

return;

}

}

MessageBox.Show("Данные введены не верно!");

}

Приложение Б

Основные процедуры android – приложения.

fun authorization(view: android.view.View) {  
 var bool: Boolean = false  
 val log = findViewById<TextInputEditText>(R.id.*logText*)  
 val pas = findViewById<TextInputEditText>(R.id.*passText*)  
 val login = log.*text*.*toString*()  
 val password = pas.*text*.*toString*()  
 val user = Users()  
 for (i in user.users.*indices*) {  
  
 if (login == user.users[i].login && password == user.users[i].password) {  
 CurrentUser.currentUser = user.users[i]  
 val intent: Intent = Intent(this, SecondActivity::class.*java*)  
 startActivity(intent)  
 bool = true  
 }  
 }  
 if (bool == false) {  
 val text = "Введён неверный логин или пароль!"  
 val duration = Toast.*LENGTH\_SHORT* val toast = Toast.makeText(*applicationContext*, text, duration)  
 toast.show()  
 }  
}

fun reg(view: android.view.View) {  
 var bool: Boolean = false  
 val pas1 = findViewById<TextInputEditText>(R.id.*pass1*)  
 val pas2 = findViewById<TextInputEditText>(R.id.*pas2*)  
 val mail = findViewById<TextInputEditText>(R.id.*email*)  
 val fio = findViewById<TextInputEditText>(R.id.*fio*)  
  
  
 if (pas1.*text*.*toString*().*isNotEmpty*() && pas2.*text*.*toString*().*isNotEmpty*() && mail.*text*.*toString*()  
 .*isNotEmpty*() && fio.*text*.*toString*().*isNotEmpty*()  
 ) {  
 bool = true  
  
 val text = "Регистрация завершена! Проверьте вашу почту!"  
 val duration = Toast.*LENGTH\_SHORT* val toast = Toast.makeText(*applicationContext*, text, duration)  
 toast.show()  
 val intent = Intent(this, SecondActivity::class.*java*)  
 intent.putExtra("pas",mail.getText().*toString*())  
 intent.putExtra("pas2",pas1.getText().*toString*())  
 intent.putExtra("pas3",fio.getText().*toString*())  
 startActivity(intent)  
 }  
 if (bool == false) {  
 val text = "Возможно вы допустили ошибку! Проверьте данные ещё раз!"  
 val duration = Toast.*LENGTH\_SHORT* val toast = Toast.makeText(*applicationContext*, text, duration)  
 toast.show()  
 }  
}

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivitySecondBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 val navView: BottomNavigationView = binding.navView  
  
 val navController = *findNavController*(R.id.*nav\_host\_fragment\_activity\_second*)  
  
 val appBarConfiguration = *AppBarConfiguration*(  
 *setOf*(  
 R.id.*navigation\_home*, R.id.*navigation\_notifications* )  
 )  
 *setupActionBarWithNavController*(navController, appBarConfiguration)  
 navView.*setupWithNavController*(navController)  
}

override fun onCreateView(  
  
 inflater: LayoutInflater,  
 container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
 notificationsViewModel =  
 ViewModelProvider(this).get(NotificationsViewModel::class.*java*)  
  
 \_binding = FragmentNotificationsBinding.inflate(inflater, container, false)  
 val root: View = binding.*root* val imageAvatar = root.findViewById<ImageView>(R.id.*avatarImage*)  
 val textName = root.findViewById<TextView>(R.id.*name*)  
 val textInfo = root.findViewById<TextView>(R.id.*info*)  
 val textOpis = root.findViewById<TextView>(R.id.*number*)  
 val view = *activity* val intent = view?.*intent* if (intent != null) {  
 textName.*text* = intent.getStringExtra("pas3")  
 textOpis.*text* = intent.getStringExtra("pas")  
 textInfo.*text* = intent.getStringExtra("pas2")  
 }  
 if (CurrentUser.currentUser != null)  
 {  
 textInfo.*text* = CurrentUser.currentUser!!.number  
 imageAvatar.setImageResource(CurrentUser.currentUser!!.avatar)  
 textName.*text* = CurrentUser.currentUser!!.nickname  
 textOpis.*text* = CurrentUser.currentUser!!.email  
 }  
  
  
 notificationsViewModel.text.observe(*viewLifecycleOwner*, *Observer* **{  
  
 }**)  
 return root  
}

override fun onCreateView(  
 inflater: LayoutInflater,  
 container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
 homeViewModel =  
 ViewModelProvider(this).get(HomeViewModel::class.*java*)  
  
 \_binding = FragmentHomeBinding.inflate(inflater, container, false)  
 val root: View = binding.*root* val users = Users()  
 val userArrayList = ArrayList<User>()  
  
 for (i in users.users.*indices* ) {  
 if (users.users[i].toys == true) {  
  
  
 userArrayList.add(users.users[i])  
 }  
 }  
 val listAdapter = ListAdapter(getActivity()?.getApplicationContext(), userArrayList)  
 val listview = root.findViewById<ListView>(R.id.*listview*)  
 listview.setAdapter(listAdapter)  
 listview.setClickable(true)  
  
  
  
 homeViewModel.text.observe(*viewLifecycleOwner*, *Observer* **{  
  
 }**)  
 return root  
}