SPRINT REVIEW II

Projekt zespołowy systemu informatycznego 2023/2024 N

Grupa: P2D4N

System POS dla baru/pubu

Scrum master: Filip Statkiewicz

Skład grupy:

- Filip Statkiewicz
- Dawid Romanów
- Michał Owsiak
- Jan Śliwak

Część 1
Sprawozdanie z przebiegu zadań

Status Produkt Backlog

Legenda:
Zadania zakończone
Zaplanowane

1. Utworzenie projektowego serwera Discord do komunikacji między sobą.
2. Utworzenie projektu zespołowego na platformie Jira.
3. Utworzenie projektu zespołowego na platformie Github.
4. Utworzenie projektu zespołowego na platformie Figma
5. Zaprojektowanie głównego okna interfejsu graficznego programu.
6. Zaprojektowanie interfejsu graficznego panelu sprzedaży.
7. Zaprojektowanie interfejsu graficznego panelu logowania i rozpoczęcia, zakończenia czasu pracy.
8. Zaprojektowanie bazy danych produktów w menu
9. Zaprojektowanie bazy danych pracowników
10. Implementacja połączenia aplikacji z bazą danych
11. Implementacja głównego okna interfejsu graficznego programu.
12. Implementacja interfejsu graficznego panelu sprzedaży.
13. Implementacja interfejsu graficznego panelu logowania i rozpoczęcia, zakończenia czasu pracy.
14. Implementacja bazy danych
15. Implementacja systemu logowania i zarządzaniem czasem pracy
16. Implementacja systemu tworzenia, edycji i usunięcia zamówienia
17. Implementacja funkcji sumującej zamówienia (cena, ilość)
18. Implementacja wyszukiwarki do produktów
19. Zaprojektowanie bazy danych przepisów
20. Implementacja możliwości użycia opcji rabatowej (np. -15%)
21. Implementacja systemu płatności
22. Implementacja systemu drukowania paragonu
23. Implementacja możliwości podglądu przepisu
24. Implementacja generowania wykazu czasu pracy pracowników
25. Implementacja funkcji historii zamówień
26. Poprawa aspektów wizualnych zaimplementowanego interfejsu graficznego Panelu Sprzedaży, Panelu Logowania i Ekranu głównego
27. Implementacja funkcjonalności TODO Listy na ekranie głównym aplikacji
28. Zaprojektowanie bazy danych listy zadań
29. Rozwiązanie problemu z Entity Frameworkiem
30. Implementacja funkcji usunięcia całego zamówienia
31. Implementacja funkcji przechowania zamówienia
32. Implementacja możliwości użycia opcji rabatowej.
33. Implementacja funkcji wyświetlania nazwy zalogowanego do panelu sprzedaży pracownika
34. Zaprojektowanie bazy danych zamówień

Status Produkt Backlog

35. Implementacja funkcji dodawania danych do faktury dla zamówienia
36. Implementacja funkcji dodawania zamówień do bazy danych (historia zamówień)
37. Implementacja Zegara i kalendarza w głównym ekranie aplikacji
38. Implementacja systemu zarządzania zapasami w magazynie
39. Implementacja funkcji filtracji
40. Implementacja systemu tworzenia, edycji, zmian przepisów
41. Implementacja systemu zarządzania produktami dostępnymi w panelu sprzedaży
42. Implementacja funkcji generowania i wyświetlania raportów sprzedaży
43. Implementacja funkcji generowanie raportu zużycia produktów
44. Implementacja funkcji sprawdzenia stanu kasy fiskalnej
45. Implementacja generowania wykazu produktywności pracowników na podstawie ilości wykonanych zamówień
46. Implementacja funkcji generowania analizy popularności produktów.
47. Implementacja funkcji tworzenia zamówień magazynowych
48. Integracja systemu magazynowego z systemem punktu sprzedaży
49. Implementacja systemu alertów i przypomnień, np. o terminie ważności produktów
50. Zaprojektowanie interfejsu graficznego funkcji magazynowych.
51. Zaprojektowanie interfejsu graficznego systemu raportów i analiz
52. Zaprojektowanie interfejsu graficznego funkcji zarządzania kadrą pracowników
53. Zaprojektowanie bazy danych stanu magazynowego
54. Zaprojektowanie bazy danych raportów i analiz
55. Implementacja interfejsu graficznego funkcji magazynowych.
56. Implementacja interfejsu graficznego systemu raportów i analiz
57. Implementacja interfejsu graficznego funkcji zarządzania kadrą pracowników
58. Implementacja funkcji zarządzania kadrą i uprawnieniami
59. Implementacja systemu zarządzania zapasami w magazynie

Ogólny przebieg prac

- Wszystkie cele sprintu zostały zrealizowane. Niestety z przyczyn niezależnych od nas (choroba jednego z członków zespołu) pojawiły się lekkie opóźnienia, a nawet obawy o możliwość nieukończenia wszystkich wyznaczonych celów sprintu. Prowadziliśmy na ten temat sporą dyskusję, aby obowiązki chorego członka przejęli inni uczestniczy projektu, jednakże po zapewnieniach, że wszystko się uda zrobić na czas właściciel opóźnionych zadań nie został zmieniony. Ostatecznie udało się zrealizować wszystkie cele.
- Praca całej reszty grupy przebiegła sprawnie i wszystko zostało dostarczone w terminie, tym razem nie popełniliśmy błędu ze sprintu 1 i nikt nie musiał czekać aż inna z osób ukończy zadanie, aby móc rozpocząć pracę nad swoim.

Napotkane Problemy

PROBLEM:

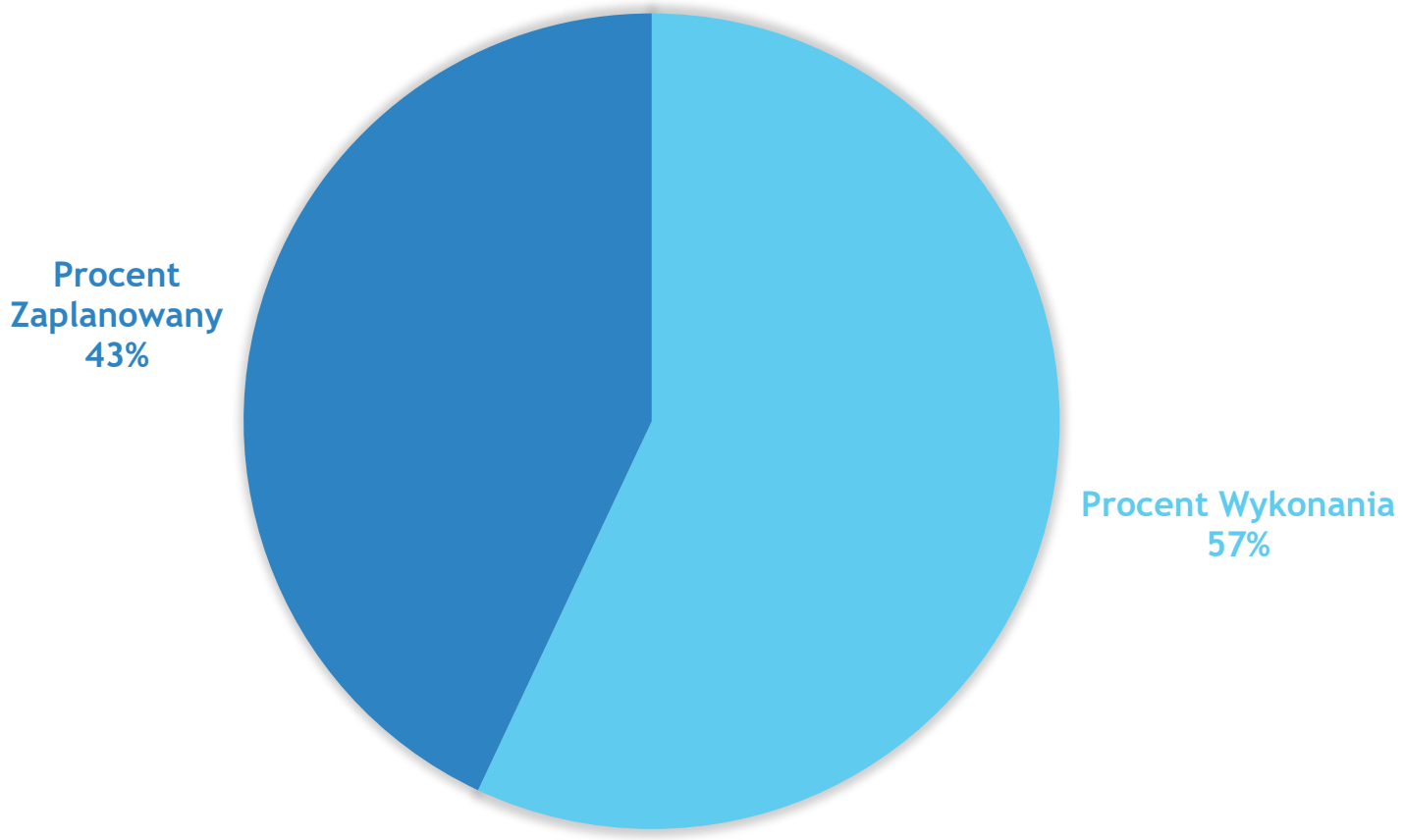
- Entity Framework i migracja danych – według konsoli menadżera pakietów wszystkie migracje przebiegały poprawnie, jednakże efektów migracji w bazie danych nie mogliśmy doświadczyć.

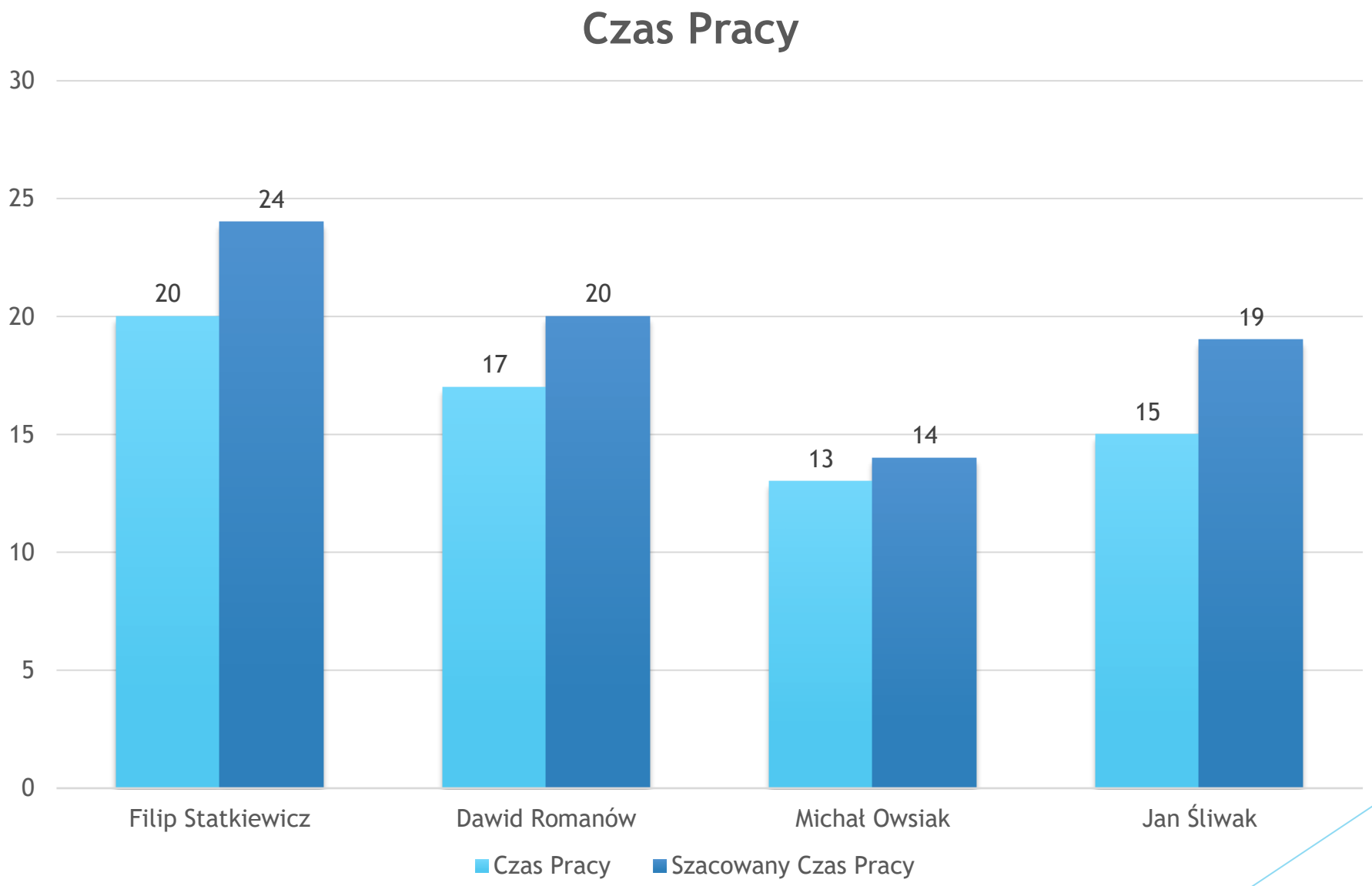
ROZWIĄZANIE

- Rozwiązanie zadania mimo iż potrzebowaliśmy na nie ok. 6 godzin okazało się być bardzo trywialne, problemem okazało się być zła ścieżka do bazy danych. Z racji użycia ścieżki relatywnej baza danych podczas wykonania migracji nie była w katalogu który zadeklarowaliśmy w kodzie, ponieważ podczas budowy lub debugowania projektu, Visual Studio tworzy inną strukturę katalogów. Zmiana ścieżki bazy danych na czas migracji rozwiązała problem.

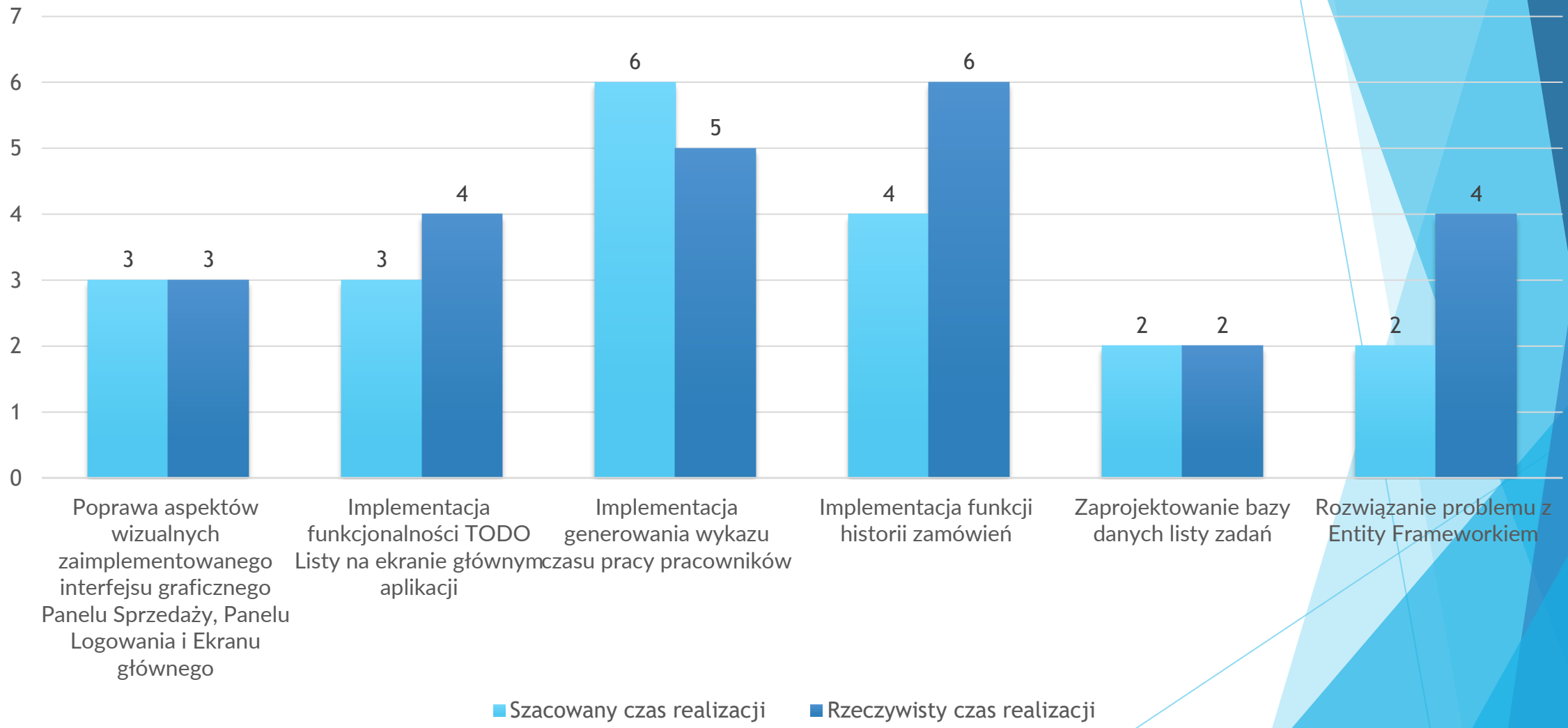
```
Odwołania: 0
static AppDbContext()
{
    //string databaseLocation = @"..\..\..\Database\barmanagement.db";
    string databaseLocation = "C:\\Users\\filip\\Programing\\C#\\POS-app-team-project\\POS\\Database\\barmanagement.db";
    string projectPath = Directory.GetCurrentDirectory();
    string absolutePath = Path.Combine(projectPath, databaseLocation);

    DatabasePath = $"Data Source=" + absolutePath;
}
```

WYKRES POKAZUJĄCY PROCENTOWY POSTĘP PRAC NAD APLIKACJĄ POS DLA BARU



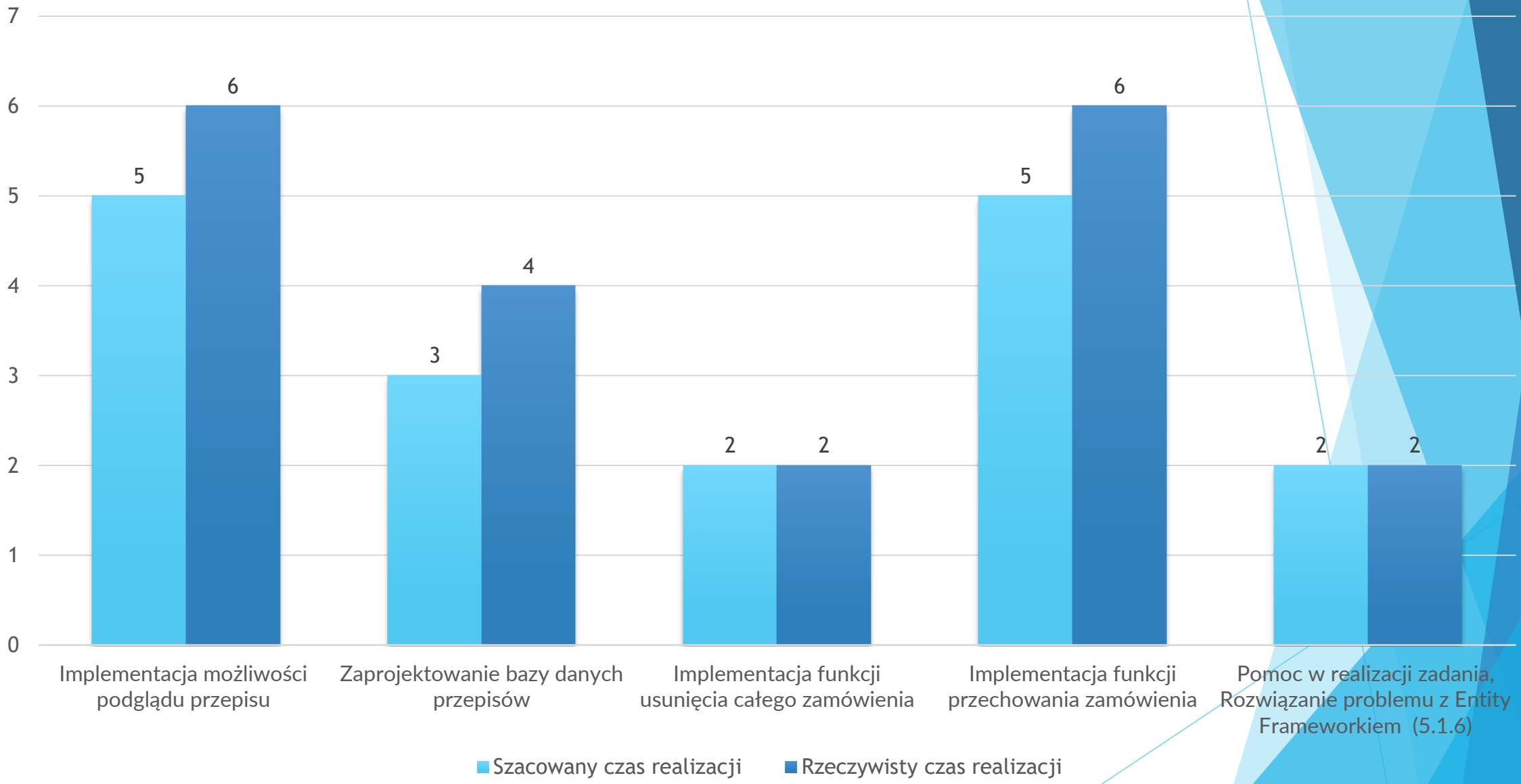

Czas Pracy
30
25
20
15
10
5
0
20
24
17
20
13
14
15
19
Filip Statkiewicz
Dawid Romanów
Michał Owsiak
Jan Śliwak
Czas Pracy
Szacowany Czas Pracy

Filip Statkiewicz



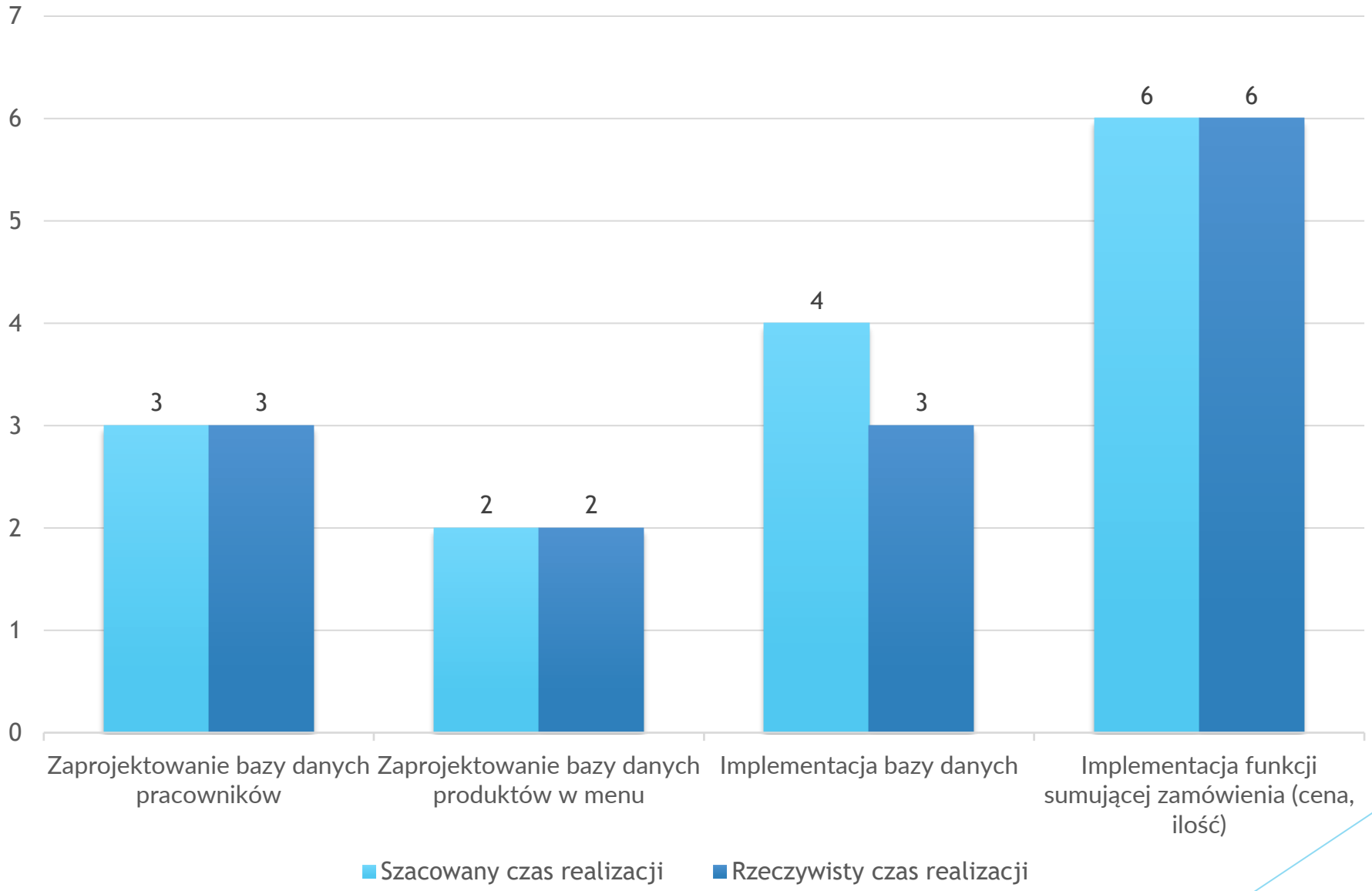
Filip Statkiewicz

Zadanie	Status	Szacowany Czas	Rzeczywisty Czas
Poprawa aspektów wizualnych zaimplementowanego interfejsu graficznego Panelu Sprzedaży, Panelu Logowania i Ekranu głównego	Ukończone	3 godziny	3 godziny
Implementacja funkcjonalności TODO Listy na ekranie głównym aplikacji	Ukończone	3 godziny	4 godziny
Implementacja generowania wykazu czasu pracy pracowników	Ukończone	6 godzin	5 godzin
Implementacja funkcji historii zamówień	Ukończone	4 godziny	6 godzin
Zaprojektowanie bazy danych listy zadań	Ukończone	2 godziny	2 godziny
Rozwiązanie problemu z Entity Frameworkiem	Ukończone	2 godziny	4 godziny


7
6
5
4
3
2
1
0
5
6
3
4
2
2
5
6
2
2
Implementacja możliwości podglądu przepisu
Zaprojektowanie bazy danych przepisów
Implementacja funkcji usunięcia całego zamówienia
Implementacja funkcji przechowania zamówienia
Pomoc w realizacji zadania, Rozwiązanie problemu z Entity Frameworkiem (5.1.6)
Szacowany czas realizacji
Rzeczywisty czas realizacji

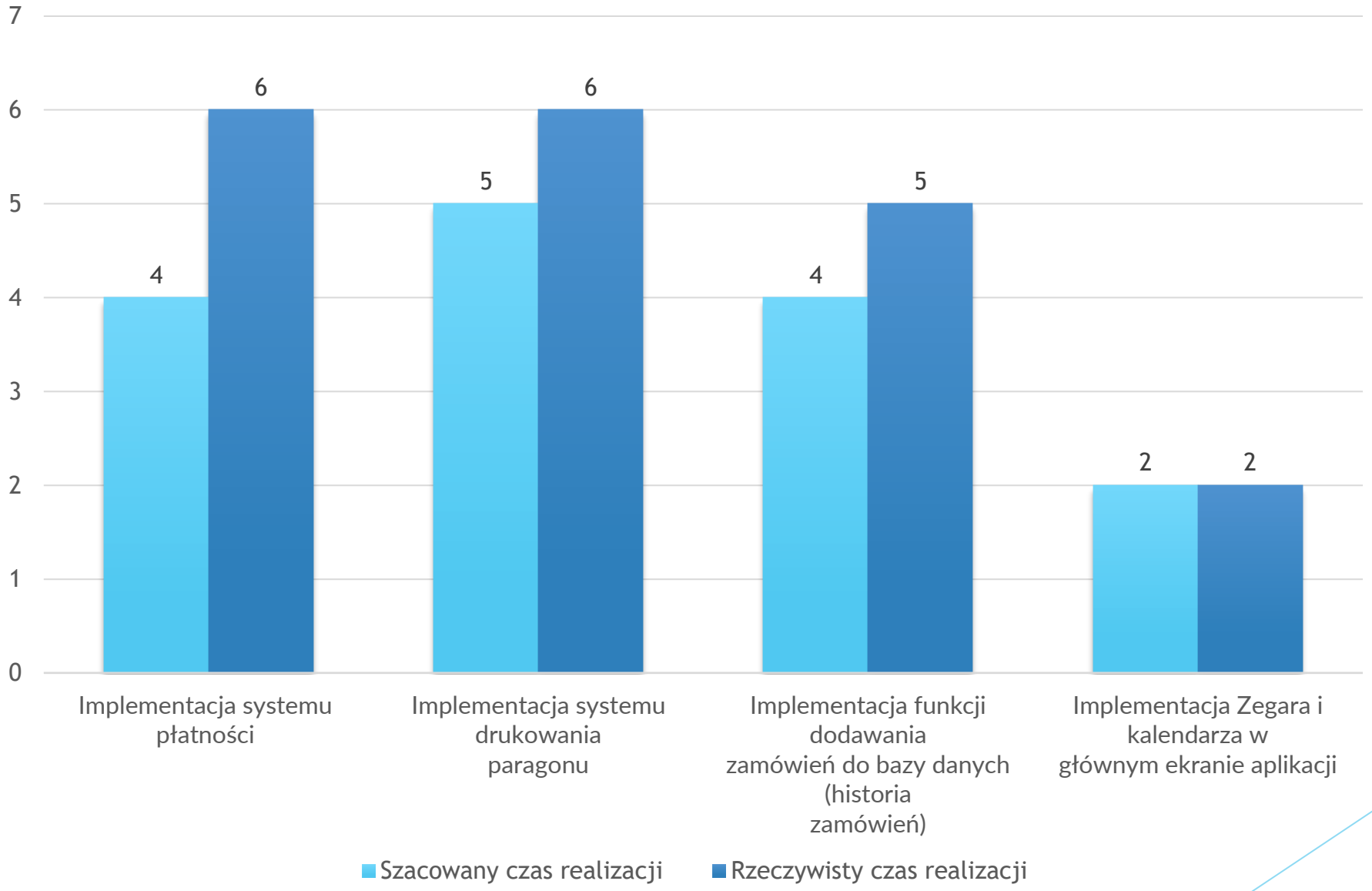
Dawid Romanów

Zadanie	Status	Szacowany Czas	Rzeczywisty Czas
Implementacja możliwości podglądu przepisu	Ukończone	5 godzin	6 godzin
Zaprojektowanie bazy danych przepisów	Ukończone	3 godziny	4 godziny
Implementacja funkcji usunięcia całego zamówienia	Ukończone	2 godziny	2 godziny
Implementacja funkcji przechowania zamówienia	Ukończone	5 godzin	6 godzin
Pomoc w realizacji zadania, Rozwiązanie problemu z Entity Frameworkiem (5.1.6)	Ukończone	2 godziny	2 godziny


7
6
5
4
3
2
1
0
3
3
2
2
4
3
6
6
Zaprojektowanie bazy danych pracowników
Zaprojektowanie bazy danych produktów w menu
Implementacja bazy danych
Implementacja funkcji sumującej zamówienia (cena, ilość)
Szacowany czas realizacji
Rzeczywisty czas realizacji

Michał Owsiak

Zadanie	Status	Szacowany Czas	Rzeczywisty Czas
Implementacja możliwości użycia opcji rabatowej.	Ukończone	3 godziny	3 godziny
Implementacja funkcji wyświetlania nazwy zalogowanego do panelu sprzedaży pracownika	Ukończone	2 godziny	2 godziny
Zaprojektowanie bazy danych zamówień	Ukończone	4 godziny	3 godziny
Implementacja funkcji dodawania danych do faktury dla zamówienia	Ukończone	4 godziny	6 godzin


7
6
5
4
3
2
1
0
4
6
5
6
4
5
2
2
Implementacja systemu płatności
Implementacja systemu drukowania paragonu
Implementacja funkcji dodawania zamówień do bazy danych (historia zamówień)
Implementacja Zegara i kalendarza w głównym ekranie aplikacji
Szacowany czas realizacji
Rzeczywisty czas realizacji

Jan Śliwak

Zadanie	Status	Szacowany Czas	Rzeczywisty Czas
Implementacja systemu płatności	Ukończone	4 godziny	6 godzin
Implementacja systemu drukowania paragonu	Ukończone	5 godzin	6 godzin
Implementacja funkcji dodawania zamówień do bazy danych (historia zamówień)	Ukończone	4 godzin	5 godzin
Implementacja Zegara i kalendarza w głównym ekranie aplikacji	Ukończone	2 godziny	2 godziny

Co udało się osiągnąć

- Możliwość dodania oraz usunięcia zadania z listy zadań w panelu głównym aplikacji
- Możliwość przeglądania otwartych oraz zamkniętych sesji pracy, czas rozpoczęcia pracy, czas zakończenia oraz ile czasu pracownik był zalogowany
- Możliwość rozpoczęcia oraz zakończania pracy w systemie przez pracownika
- Możliwość śledzenia daty i czasu w systemie (z racji tego, że nasz aplikacja wymusza na użytkowniku korzystania z trybu pełnoekranowego)
- Możliwość przeglądania przepisów na wybrane produkty dostępne w panelu sprzedaży
- Możliwość przechowania otwartego zamówienia i rozpoczęcie kolejnego, oraz możliwość powrotu w dowolnym momencie do wcześniej otwartego zamówienia

Co udało się osiągnąć

- Możliwość użycia rabatu na zamówienie
- Możliwość dodania danych do faktury
- Możliwość opłacenia zamówienia kartą / gotówką wraz z wygenerowaniem imitacji paragonu (w formacie pdf)
- Możliwość przeglądania historii zamówień
- Możliwość podglądu aktualnie zalogowanego pracownika do panelu sprzedaży

Możliwość przeglądania otwartych oraz zamkniętych sesji pracy, czas rozpoczęcia pracy, czas zakończenia oraz ile czasu pracownik był zalogowany

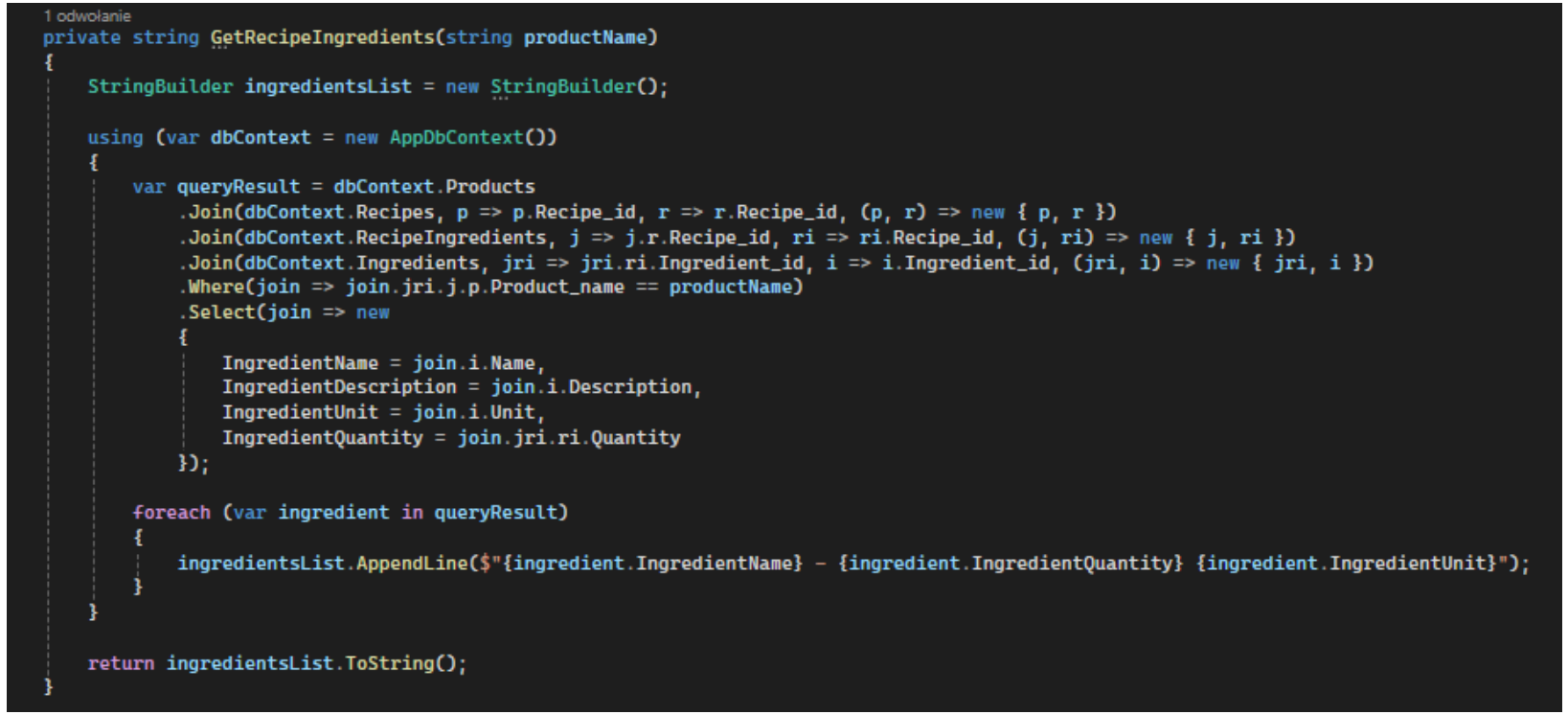
```
Odwołania: 3
private void ShowActiveSessions()
{
    ActiveSessions.Clear();
    using (var dbContext = new AppDbContext())
    {
        var employeeWorkSession = dbContext.EmployeeWorkSession.ToList();

        if (employeeWorkSession != null)
        {
            foreach (var session in employeeWorkSession)
            {
                var user = dbContext.Employees.FirstOrDefault(e => e.Employee_id == session.Employee_Id);
                if (user != null)
                {
                    DateTime workingTimeFrom = DateTime.ParseExact(session.Working_Time_From, "HH:mm", CultureInfo.InvariantCulture);
                    DateTime workingTimeTo;

                    if (session.Working_Time_To == "" || session.Working_Time_To == null)
                    {
                        workingTimeTo = DateTime.Now;
                    }
                    else
                    {
                        workingTimeTo = DateTime.ParseExact(session.Working_Time_To, "HH:mm", CultureInfo.InvariantCulture);
                    }

                    TimeSpan workingTimeDifference = (workingTimeTo - workingTimeFrom);
                    byte hours = (byte)workingTimeDifference.TotalHours;
                    byte minutes = (byte)workingTimeDifference.Minutes;
                    string formattedTimeDifference = $"{hours:D2}:{minutes:D2}";

                    session.Working_Time_Summary = formattedTimeDifference;
                    dbContext.SaveChangesAsync();
                    ActiveSessions.Add(session);
                }
            }
        }
    }

    workingTimeSummaryDataGrid.ItemsSource = ActiveSessions;
}
```

Rys. 1

Na rys. 1 przedstawiony jest kod odpowiedzialny za generowanie listy aktywności użytkowników w systemie. Kod pobiera dane z bazy danych oraz formatuje je w sposób czytelny dla człowieka, aby później wygenerować tabelę z przedstawionymi danymi.

Autor: Filip Statkiewicz

Możliwość przeglądania przepisów na wybrane produkty dostępne w panelu sprzedaży



```
1 odwołanie
private string GetRecipeIngredients(string productName)
{
    StringBuilder ingredientsList = new StringBuilder();

    using (var dbContext = new AppDbContext())
    {
        var queryResult = dbContext.Products
            .Join(dbContext.Recipes, p => p.Recipe_id, r => r.Recipe_id, (p, r) => new { p, r })
            .Join(dbContext.RecipeIngredients, j => j.r.Recipe_id, ri => ri.Recipe_id, (j, ri) => new { j, ri })
            .Join(dbContext.Ingredients, jri => jri.ri.Ingredient_id, i => i.Ingredient_id, (jri, i) => new { jri, i })
            .Where(join => join.jri.j.p.Product_name == productName)
            .Select(join => new
            {
                IngredientName = join.i.Name,
                IngredientDescription = join.i.Description,
                IngredientUnit = join.i.Unit,
                IngredientQuantity = join.jri.ri.Quantity
            });

        foreach (var ingredient in queryResult)
        {
            ingredientsList.AppendLine($"{ingredient.IngredientName} - {ingredient.IngredientQuantity} {ingredient.IngredientUnit}");
        }
    }

    return ingredientsList.ToString();
}
```

Rys. 2

Autor: Dawid Romanów

Na rys. 2 przedstawiony jest kod odpowiedzialny za generowanie listy potrzebnych składników do wykonania danego produktu znajdującego się w panelu sprzedaży. Kod ten wykonuje skomplikowaną operację wyszukiwania w bazie danych aby następnie znalezione dane umieścić w liście i zwrócić ją jako wynik działania funkcji

Możliwość użycia rabatu na zamówienie

```
1 odwołanie
private void ApplyDiscount_Click(object sender, RoutedEventArgs e)
{
    if (discountApplied)
    {
        MessageBox.Show("Rabat został już zastosowany.", "Informacja", MessageBoxButton.OK, MessageBoxImage.Information);
        return;
    }
    DiscountWindow discountWindow = new DiscountWindow();
    discountWindow.Owner = this;

    if (discountWindow.ShowDialog() == true)
    {
        double discountRate = discountWindow.radioButton10.IsChecked == true ? 0.1 : 0.15;

        foreach (var orderItem in orderListCollection[currentOrderId])
        {
            orderItem.Price *= (1 - discountRate);
        }
        UpdateTotalPrice();
        discountApplied = true;
    }
}
```

Rys. 3

Autor: Michał Owsiak

Rys. 3 Przedstawia kod odpowiedzialny za możliwość użycia rabatu na zamówienie. Kod ten najpierw sprawdza czy nie został wprowadzony już jakiś rabat następnie wyświetla okienko dialogowe z możliwością wybrania opcji rabatu. Na koniec wylicza nową cenę zamówienia po rabacie i aktualizuję cenę zamówienia.

Możliwość opłacenia zamówienia kartą / gotówką wraz z wygenerowaniem imitacji paragonu (w formacie pdf)

```
Odwołania: 2
private void PayForOrder_Click(object sender, RoutedEventArgs e)
{

    if (sender is Button button && button.Tag is string paymentMethod)
    {
        double totalPrice = Math.Round(orderList.Sum(item => item.Amount * item.Price), 2);
        GenerateBill printWindow = new GenerateBill(orderList);
        printWindow.ShowDialog();
        var order = SaveOrder();
        SaveOrderItems(order);
        SavePayment(order, paymentMethod, totalPrice);
        orderList.Clear();
        UpdateTotalPrice();
        MessageBox.Show($"Zapłacono za zamówienie {totalPrice:C} - metoda płatności: {paymentMethod}");
    }

}
```

Rys. 4

Autor: Jan Śliwak

Rys. 4 przedstawia kod odpowiedzialny za całą logikę kryjącą się pod przyciskiem zapłaty za zamówienie. Kod ten oblicza wartość zamówienia, generuje dokument rachunek (pdf), zapisuje zamówienie w bazie danych, zapisuje płatność w bazie danych, oraz aktualizuje panel sprzedaży aby doprowadzić go do stan początkowego i gotowości rozpoczęcia nowego zamówienia

Część 2
Prezentacja działania aplikacji

https://www.youtube.com/watch?v=I5xFZh08FHk