Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki

Aplikacje Mobilne - Projekt

Rok akademicki - 2024/2025

Temat projektu: Aplikacja do Treningu	**Wykonali:** Stępień Filip Karwat Damian Grot Rafał **Grupa:** 3ID11B

SPIS TREŚCI

1. Wstęp

a. Krótki opis aplikacji

Zadaniem stworzonej aplikacji jest asystowanie w śledzeniu postępów, oraz możliwość prowadzenia dziennika dla treningów na siłowni. Aplikacja udostępnia poniżej wymienione funkcjonalności:

- Logowanie i Rejestracja,
  - Użytkownik może stworzyć własne konto, lub zalogować się na istniejące.
  - Do każdego konta przypisane są dane użytkownika, takie jak statystyki, poprzednio wykonane treningi, oraz dane logowania.
  - Aplikacja posiada funkcje szybkiego logowania się – logowanie następuje automatyczne, jeśli na danym urządzeniu ktoś był zalogowany wcześniej.
- Edycja danych logowania,
  - Użytkownik może zmienić swój login i hasło.
- Śledzenie statystyk dotyczących ćwiczeń i treningów,
  - Na profilu użytkownika dostępne są statystyki z wykonanych treningów,
  - Na ekranie głównym znajduje się lista poprzednich treningów, zawierająca ich szczegóły.
- Przeglądanie predefiniowanych ćwiczeń,
  - Na ekranie ćwiczeń znajduję się lista ćwiczeń dostępnych w aplikacji, każde ćwiczenie ma przypisane mięśnie na które oddziałuje, oraz instrukcje które można przeczytać po kliknięciu.
- Tworzenie treningu z gotowych ćwiczeń,
  - Podczas treningu użytkownik może dodawać do niego dowolną ilość ćwiczeń z biblioteki ćwiczeń,

    - Dla każdego wykonywanego ćwiczenia użytkownik dodaje ilość serii, powtórzeń, oraz wagę jakiej użył.

- Zapisywanie treningów w kalendarzu.
    - Treningi zapisywane są w kalendarzu ćwiczeń, umożliwiając użytkownikowi późniejsze podejrzenie w jakich dniach trenował, oraz ile dni odpoczywał, bądź ile dni z rzędu odbył trening.

b. Wykorzystana technologia i narzędzia

Do stworzenia aplikacji wykorzystano następujące technologie i narzędzia:

- **Środowisko Expo** w wersji ~52.0.4 - umożliwia łatwą konfigurację i uruchamianie aplikacji mobilnych w oparciu o React Native, oraz udostępnia wiele dodatkowych narzędzi i interfejsów, takich jak expo-secure-store.

- **Język programowania TypeScript** w wersji ^5.3.3, oraz jego biblioteka React w wersji 18.3.1,. Do tworzenia aplikacji mobilnej wykorzystano **React Native** w wersji 0.76.1,

- **React Navigation** w wersjach 7.0.0, (Stack Navigation) oraz 7.0.1 (Bottom Tab Navigation), który odpowiada za zarządzanie nawigacją między ekranami aplikacji.

- **Biblioteka React Native Reanimated** w wersji ~3.16.1, używana do tworzenia płynnych animacji w interfejsie użytkownika. W aplikacji użyta do ustawienia motywów, oraz do tworzenia tabel.

- **Expo Image Picker** w wersji ~16.0.3, pozwalający na wybieranie obrazów z galerii urządzenia lub robienie zdjęć aparatem. W aplikacji użyta do dodawania zdjęć przypisanych do danych treningów.

- **Expo Secure Store** w wersji ~14.0.0, wykorzystywany do bezpiecznego przechowywania danych użytkownika.

- **Axios** w wersji ^1.7.8, który służy do obsługi zapytań HTTP i komunikacji z API.

- **Day.js** w wersji ^1.11.13, lekka biblioteka do manipulacji i formatowania dat przypisanych do treningów,
- **Victory Native** w wersji ^41.14.0, jest to biblioteka służąca do tworzenia wykresów, w tym przypadku wykresu kołowego i liniowego,
- **React Native Skia** w wersji ^1.7.3, biblioteka służąca do generowania grafik 2d w React-Native, używana przy tworzeniu wykresu rozkładu treningów
- **Fuse.js** w wersji ^7.0.0, do implementacji wyszukiwania w czasie rzeczywistym, użyta do inteligentnego wyszukiwana ćwiczeń,
- **React Native Paper** w wersji ^5.12.5, która dostarcza gotowe komponenty UI zgodne z Material Design, zostały użyte z niej komponenty jak i motywy,
- **Json-server** w wersji ^1.0.0-beta.3, używany do tworzenia lokalnego serwera API w celach testowych.
- **Narzędzia ESLint** (^8.57.0) i **Prettier** (^3.3.3), które zapewniają jednolitą jakość i formatowanie kodu.
- **Babel** w wersji ^7.25.2, służący do transpilacji nowoczesnego JavaScriptu na kod kompatybilny z urządzeniami mobilnymi.

c. Wykorzystane natywne funkcje urządzenia

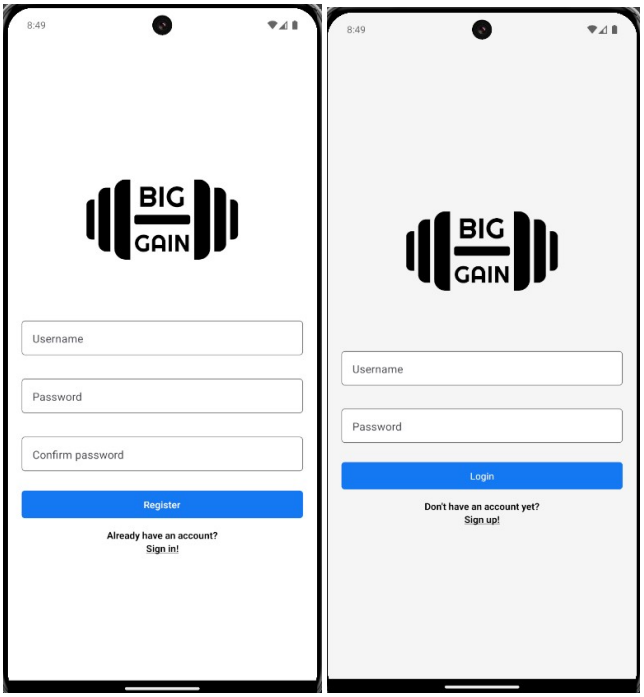
W aplikacji wykorzystano następujące funkcje natywne urządzenia:

- Aparat - **Expo Image Picker –** umożliwia użytkownikowi aplikacji dodawanie zdjęć z treningu, i zapisywanie go.
- Wibracje – **React Native Vibration –** wibracje urządzenia uruchamiają się, gdy dana część treningu się skończy, w formie przypomnienia, że należy kontynuować trening.

2. Implementacja

a. Ekrany dostępne w aplikacji

- Logowanie i rejestracja



Na ekranie po lewej stronie widać jest pola związane z rejestracją użytkownika, takie jak Username, Password, i Confirm Password. Poniżej znajduje się przycisk potwierdzający rejestrację, i opcja do przełączenia na ekran logowania, który widać po prawej stronie. Na nim widnieją tylko pola do wpisania nazwy użytkownika i hasła, oraz przyciski do potwierdzenia logowania i przełączenia na ekran rejestracji.

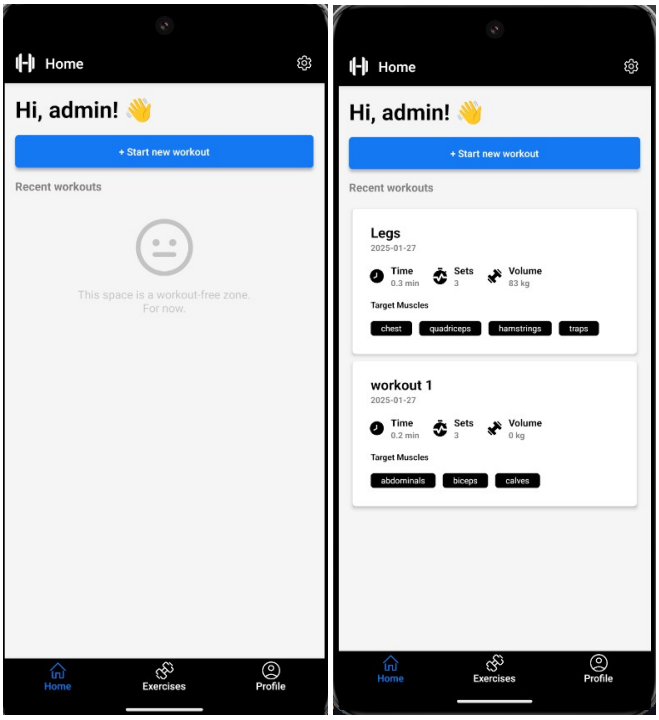
```
const handleLogin = async ({ username, password }: Credentials) => {
    setIsLoginPending(true);
    authenticate({ username, password })
        .then(id => {
            console.log(id);
            if (id === null) {
                setLoginFailed('Incorrect username or password');
            } else {
                setUserData({ id, username });
            }
        })
        .catch(e => {
            console.error('Error during login ' + e);
            setLoginFailed('' + e);
        })
        .finally(() => {
            setIsLoginPending(false);
        });
};
```

Funkcja autoryzująca logowanie użytkownika

```
useEffect(() => {
    getCredentials().then((e: Credentials) => {
        console.log('saved creds' + JSON.stringify(e));
        if (e?.password !== undefined && e?.username !== undefined) {
            handleLogin(e).finally(() => {
                setUsernameFailed('');
                setLoginFailed('');
            });
        }
    });
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);
```

Funckja wykonująca autoamtyczne logowanie

- Ekran Główny

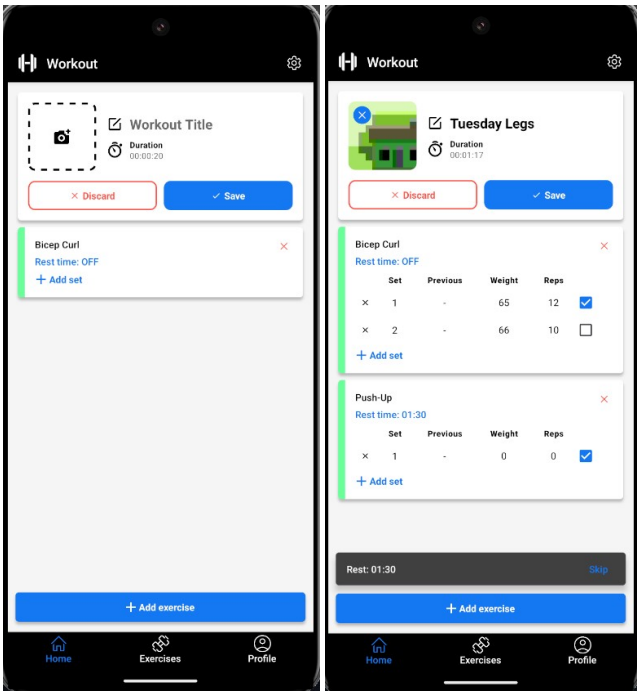


Po zalogowaniu się, aplikacja przenosi nas na ekran główny, gdzie znajdują się: przycisk przenoszący do ustawień w prawym górnym rogu, przycisk do rozpoczęcia treningu, dolna belka nawigacji, oraz jeśli istnieją, to lista poprzednich treningów.

```
useFocusEffect(
    useCallback(() => {
        const fetchWorkouts = async () => {
            const data = await getWorkouts(userID);
            setWorkouts(data.reverse());
        };

        fetchWorkouts();
    }, [userID])
);
```

Funkcja pobierająca poprzednie treningi na podstawie id użytkownika

- Aktualny Trening



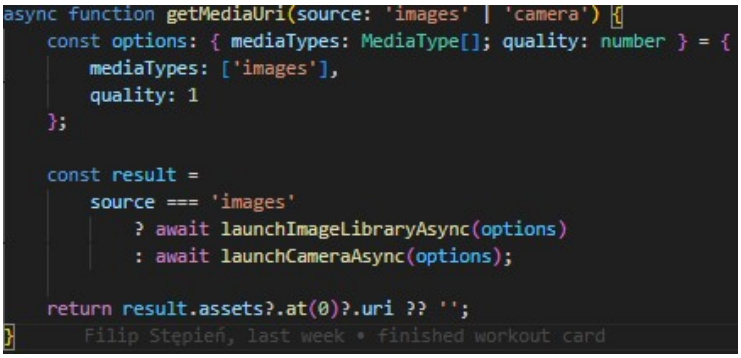
Na tym ekranie uzytkownik przeprowadza swój trening. W górnej części można dodać zdjęcie, ustawić tytuł, i zakończyć lub usunąć bieżący trening. Niżej wyświetlana jest lista ćwiczeń wykonywanych na tym treningu. Na karcie każdego ćwiczenia można ustawić czas odpoczynku, dodawać i usuwać serie, a w nich ustawiać ilosc powtórzeń oraz wagę. Gdy użytkownik zaznaczy serię jako wykonaną, na dole pojawia się powiadomienie odliczające czas odpoczynku, po którymn następują wibracje przypominające o kontynuowaniu treningu. Przycisk na samym dole służy do dodawania ćwiczeń do treningu. Ekran ten jest też używany do edycji wcześniej wykonanego treningu, dlatego jest to ekran z najbardziej złożonym kodem.

```
useEffect(() => {
    const interval = setInterval(() => {
        if (restTimeSeconds !== null && restTimeSeconds > 0) {
            const newValue = restTimeSeconds - 1;
            setRestTimeSeconds(newValue);

            if (newValue === 0) {
                setRestSnackbarVisible(false);
                clearInterval(interval);
                Vibration.vibrate(500);
            }
        }
    }, 1000);

    return () => clearInterval(interval);
}, [restTimeSeconds]);
```

Funkcja odliczająca czas odpoczynku i wywołująca wibracje

```
const saveExercises = async () => {
    const saveResult = await saveWorkout();
    if (!saveResult) return null;

    const workoutId = editMode ? (saveResult as Workout).id : (saveResult as string);

    const payload = exercises
        .filter(e => e.rows.length > 0)
        .map(e => ({
            exercise: {
                ...(editMode && { id: e.id?.split('_').at(0) as string }),
                workoutId: workoutId,
                primaryMuscle: e.exercise.primaryMuscle,
                name: e.exercise.name,
                level: e.exercise.level
            },
            rows: e.rows
        }));

    return Promise.all(
        payload.map(async p => ({
            exercise: editMode
                ? await putExercise(p.exercise.id, p.exercise as WorkoutExercise)
                : await postExercise(p.exercise),
            rows: p.rows
        }))
    );
};
```

Funkcja zapisująca wykonane ćwiczenia do API



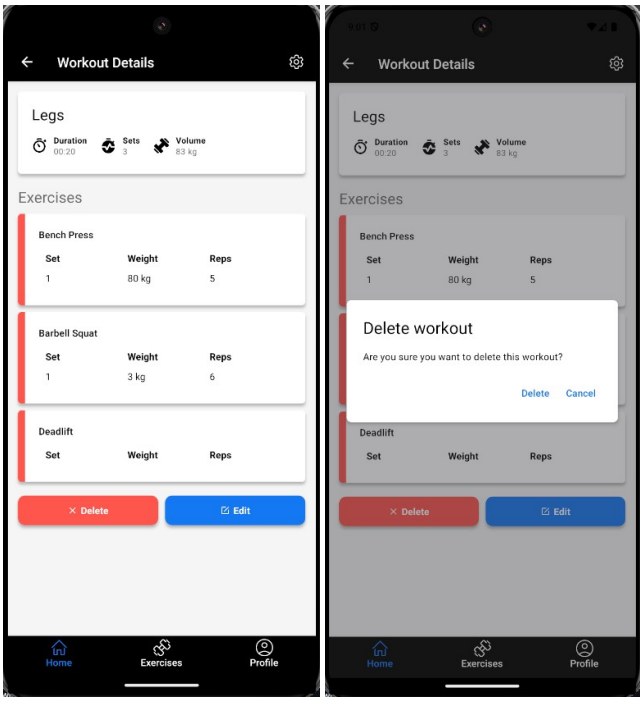
```
async function getMediaUri(source: 'images' | 'camera') {
    const options: { mediaTypes: MediaType[]; quality: number } = {
        mediaTypes: ['images'],
        quality: 1
    };

    const result =
        source === 'images'
            ? await launchImageLibraryAsync(options)
            : await launchCameraAsync(options);

    return result.assets?.at(0)?.uri ?? '';
}       Filip Stępień, last week • finished workout card
```

Funkcja pozwalająca zrobić zdjęcie lub wybrać zdjęcie z urządzenia

```
const handleAddSet = async () => {
    const newSetNumber = cardExercise.rows.length + 1;
    const prevSet = await getPrevSet(userData.id, cardExercise.exercise.name, newSetNumber);
    const lastSet = cardExercise.rows.at(-1);
    const newRow: ExerciseTableRow = {
        setNumber: newSetNumber,
        weight: null,
        reps: null,
        checked: false,
        prevReps: prevSet?.reps,
        prevWeight: prevSet?.weight,
        weightPlaceholder:
            (newSetNumber === 1 ? prevSet?.weight : lastSet?.weight) ??
            lastSet?.weightPlaceholder ??
            0,
        repsPlaceholder:
            (newSetNumber === 1 ? prevSet?.reps : lastSet?.reps) ??
            lastSet?.repsPlaceholder ??
            0
    };

    setOrAppendUserSets(newRow);
};
```

Funkcja obsługująca dodanie serii do danego ćwiczenia

- Wykonany Trening



Ten ekran pozwala na podgląd szczegółów dotyczących treningu, posiada też przyciski do usuwania treningu oraz edytowania jego danych. Ten ekran jest bazowany na ekranie treningu.

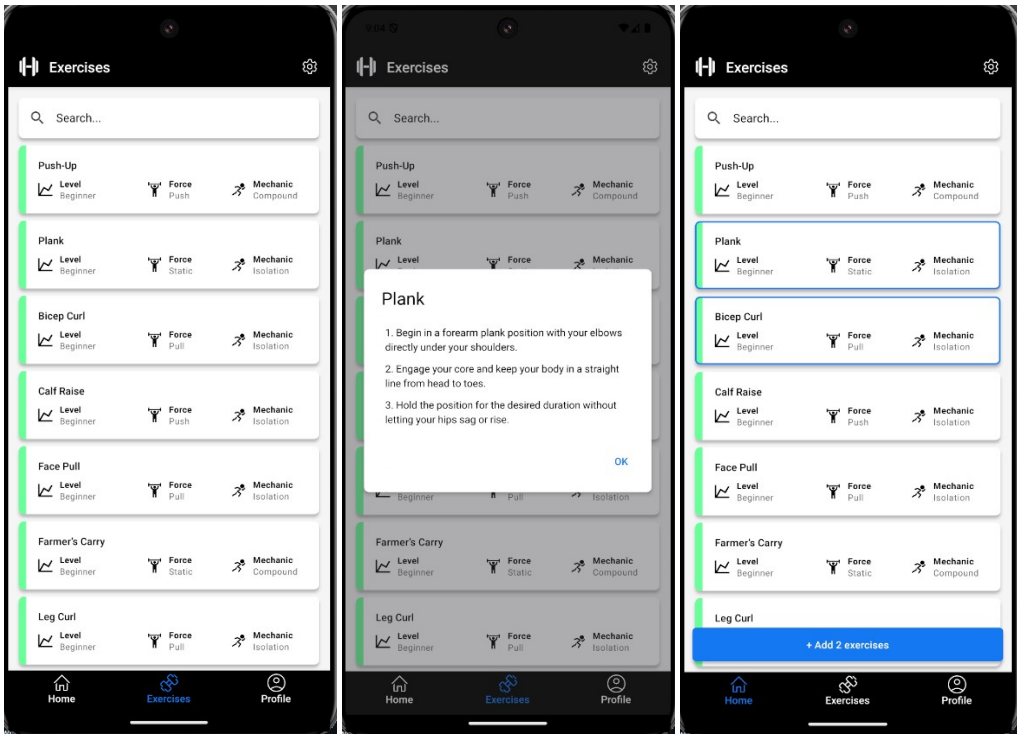
```
const saveWorkout = () => {
    const payload: Omit<Workout, 'id'> = {
        userId: userData.id as string,
        title: title || 'Untitled workout',
        imageUrl: imageUri || null,
        dateTimestamp: Math.floor(
            (editMode ? (workout?.dateTimestamp as number) : Date.now()) / 1000
        ),
        totalDuration: duration,
        totalSets: getTotalSets(exercises),
        totalVolume: getTotalVolume(exercises),
        targetMuscles: getTargetMuscles(exercises)
    };
    return editMode ? putWorkout(workout?.id, payload) : postWorkout(payload);
};
```

Funckja zapisująca trening w API

```
{!editMode && (
    <ButtonWithIcon
        iconSource={require('@assets/icons/add.png')}
        label='Add exercise'
        outlineColor={theme.colors.primary}
        color={theme.colors.onPrimary}
        backgroundColor={theme.colors.primary}
        onPress={handleExerciseAdd}
        style={{
            ...styles.addExerciseButton,
            bottom: theme.screenPadding,
            boxShadow: theme.shadowPrimary
        }}
    />
)}
```

Przycisk wyświetlany w zależności od tego czy jesteśmy w trybie edycji danych treningu

- Ćwiczenia



Ten ekran pozwala użytkownikowi na przeglądanie dostępnych ćwiczeń, oraz czytanie ich opisów/instrukcji. Po prawej stronie przedstawiona jest wersja tego ekranu, wchodząca w interakcje z ekranem treningu, gdzie użytkiownik wybiera ćwiczenia które chce wykonać, i po kliknięciu przycisku wraca na ekran treningu, gdzie ćwiczenia zostają dodane.

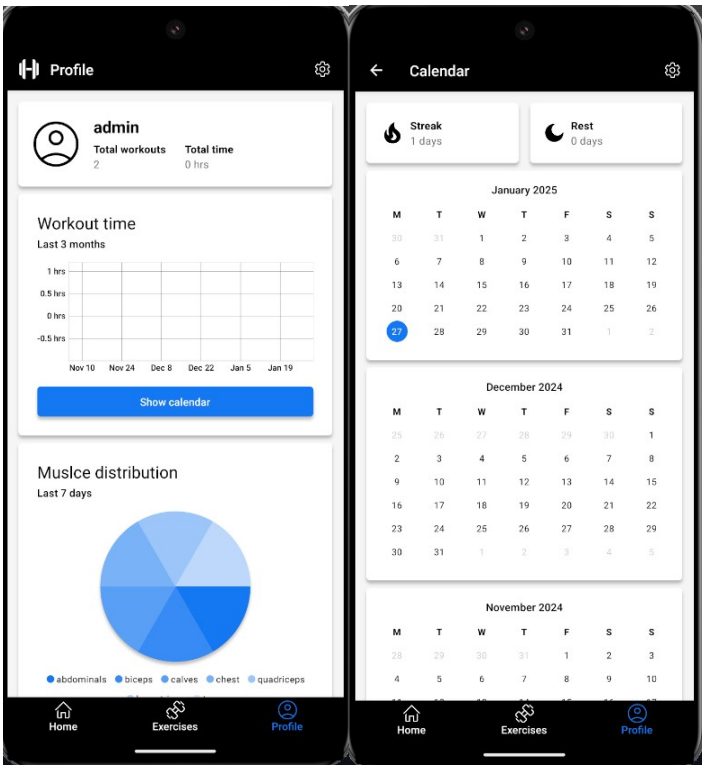
```
useEffect(() => {
    const fetchExercises = async () => {
        const exercises = await getPredefinedExercises();
        dispatch({ type: ActionType.FETCH, payload: { exercises } });
    };

    fetchExercises();
}, []);
```

Funckja pobierająca ćwiczenia z API

```
const handleSelectConfirm = () => {
    const exercises = userData.workout?.exercises
        ? [
            ...userData.workout?.exercises,
            ...state.selectedExercises.map(toWorkoutScreenExercise)
          ]
        : state.selectedExercises.map(toWorkoutScreenExercise);

    setUserData({
        ...userData,
        workout: {
            ...userData.workout,
            exercises: exercises
        }
    });

    navigation.navigate('Workout');
};
```

Funkcja obsługująca wybór ćwiczeń do treningu

- Profil ze statystykami, kalendarz



Na profilu u samej góry znajdują się podstawowe dane użytkownika. Poniżej widać tygodniowy wykres treningów (w tym przypadku pusty). Pod nim znajduje się przycisk przenoszący na ekran kalendarzam gdzie można zobaczyć w jakie dni odbywało się trening. Pod Przyciskiem znajduje się wykres pokazujący rozłożenie obciążenia mięśni na treningach.

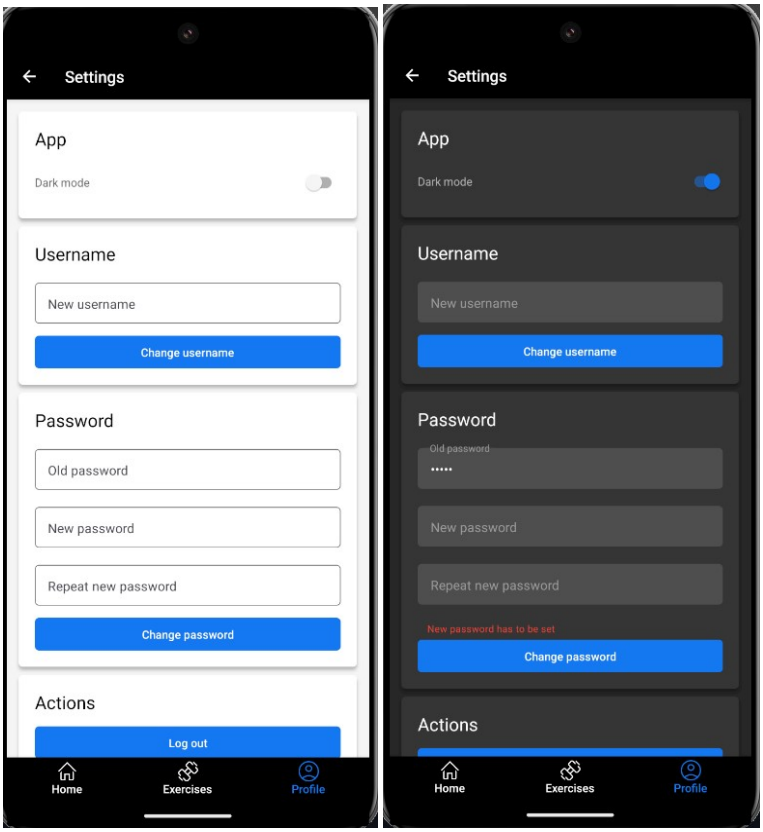
```
function getMuscleChartData(
    workouts: Workout[],
    startColor: HexColor,
    endColor: HexColor
): MuscleData[] {
    const now = dayjs();

    const lastWorkouts = workouts.filter(workout => {
        const workoutDate = dayjs.unix(workout.dateTimestamp);
        return workoutDate.isAfter(now.subtract(7, 'days'));
    });

    const muscleData: { label: string; value: number }[] = [];

    lastWorkouts.forEach(workout => {
        workout.targetMuscles.forEach(muscle => {
            const existingMuscle = muscleData.find(item => item.label === muscle.muscleName);

            if (existingMuscle) {
                existingMuscle.value += muscle.numberOfSets;
            } else {
                muscleData.push({
                    label: muscle.muscleName,
                    value: muscle.numberOfSets
                });
            }
        });
    });

    const colors = getColorGradientPalette(startColor, endColor, muscleData.length + 1);
    return muscleData.map((data, i) => ({ ...data, color: colors[i] }));
}
```

Funkcja pobierająca dane do wykresu mięśniowego\

```
function getWeeklyWorkoutData(workouts: Workout[]): WorkoutData[] {
    const now = dayjs();
    const weeksCount = 12;
    const weeks: { start: dayjs.Dayjs; totalDuration: number }[] = [];

    for (let i = 0; i < weeksCount; i++) {
        const startOfWeek = now.subtract(i, 'week').startOf('week');
        weeks.unshift({ start: startOfWeek, totalDuration: 0 });
    }

    workouts.forEach(workout => {
        const workoutDate = dayjs.unix(workout.dateTimestamp);
        weeks.forEach(week => {
            if (workoutDate.isSame(week.start, 'week')) {
                week.totalDuration += workout.totalDuration;
            }
        });
    });

    return weeks.map(week => ({
        week: week.start.format('MMM D'),
        totalHours: Math.round(week.totalDuration / 3600)
    }));
}
```

Funckja pobierająca i przypisująca dane do wykresu treningów

- Ustawienia



Na ekranie ustawień pierwsza możliwość to zmiana motywu aplikacji. Poniżej użytkownik może zmienić dane logowania (nazwę i hasło), i na samym dole jest możliwość wylogowania się z aplikacji. Po kliknięciu przycisku następuje wylogowanie i przeniesienie na ekran logowania.

```
useEffect(() => {
    if (isFirstRender.current) {
        isFirstRender.current = false;
        return;
    }

    if (swtichToggled) {
        changeTheme(darkTheme, { darkMode: true });
    } else {
        changeTheme(lightTheme);
    }
}, [swtichToggled, changeTheme, isFirstRender]);

const onToggleDarkmodeSwitch = () => setSwtichToggled(prev => !prev);
```

Funkcja renderująca elementy aplikacji w zależności od wybranego motywu.

```
const onLogout = () => {
    setLogoutIndicator(true);
    logout()
        .then(() => setUserData({}))
        .finally(() => {
            setLogoutIndicator(false);
        });
};
```

Funkcja obsługująca wylogowanie

```
changePassword(userData.id, password, newPassword)
    .then(() => {
        console.log('Password changed');
        ToastAndroid.show('Password changed', 2000);

        const curUser = userData.username;
        if (curUser === undefined) {
            throw new Error('Current user not found');
        }
        // to update local storage data
        saveCredentialsAsync({ username: curUser, password: newPassword })
            .then(e => {
                console.log('updated');
            })
            .catch(e => {
                console.error(e);
            });
    })
    .catch(e => {
        ToastAndroid.show(e.message, 5000);
    })
    .finally(() => {
        setIsPasswordChangePending(false);
    });
```

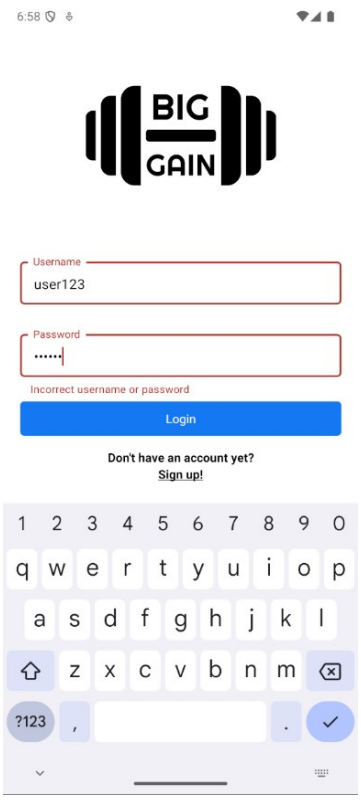
Funkcja obsługująca zmianę hasła użytkownika

b. Link do repozytorium

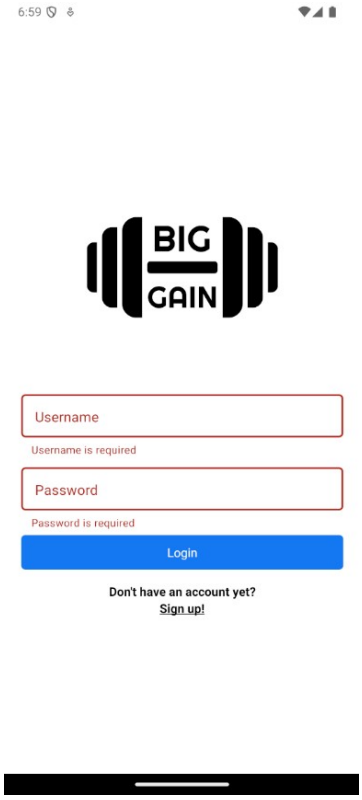
Poniżej znajduje się link do zdalnego repozytorium gdzie można zobaczyć kod całej aplikacji:

https://github.com/kurczakooo/React-Native-Project

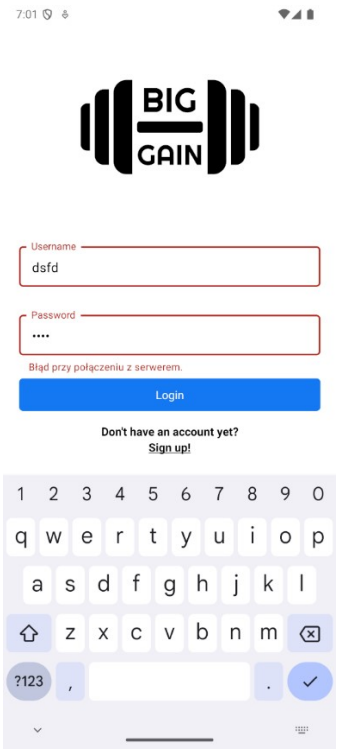
3. Testy aplikacji



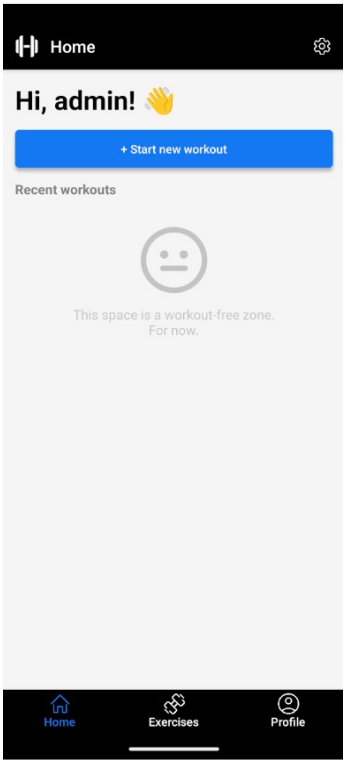
Wpisanie niepoprawnych danych logowania



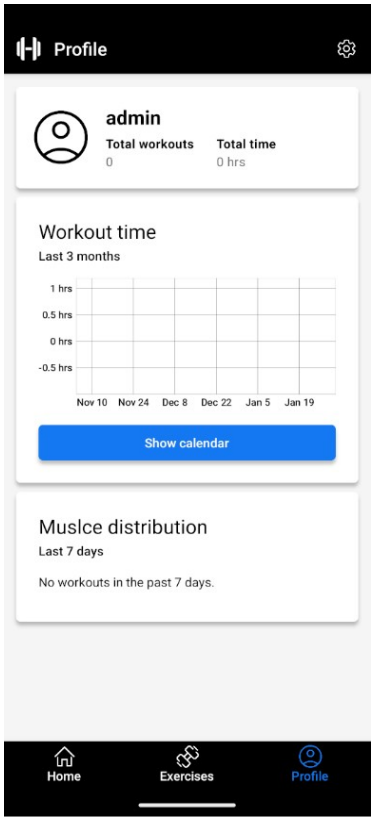
Wpisanie pustych danych logowania



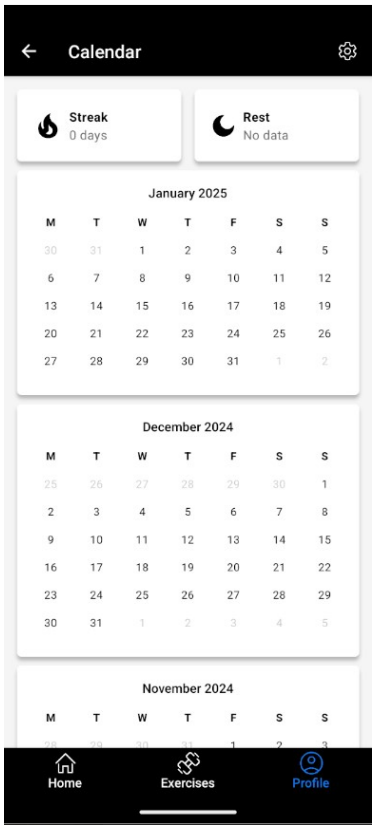
Błąd połączenia z API



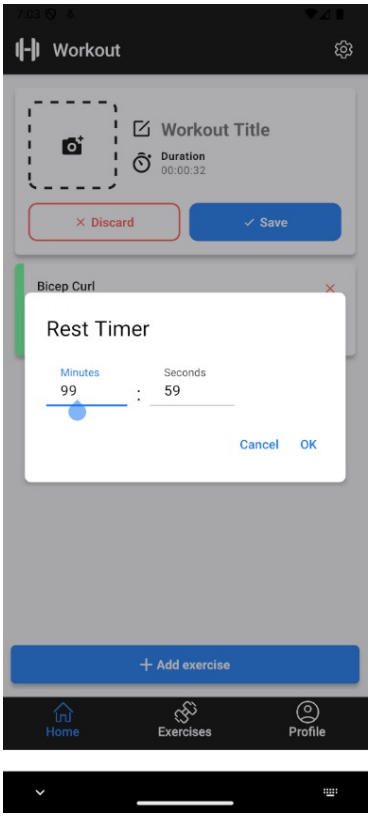
Brak treningów do wyświetlenia



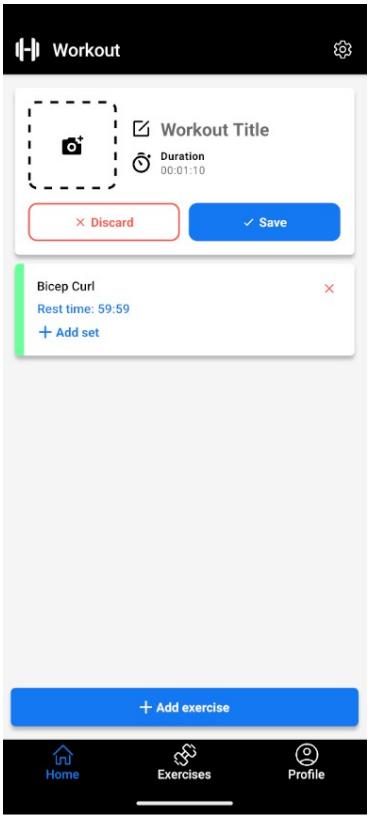
Brak statystyk do wyświetlenia (profil)



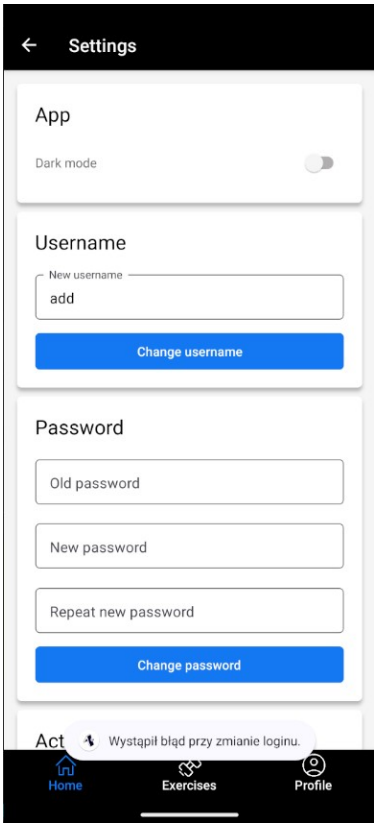
Brak statystyk do wyświetlenia (kalendarz)



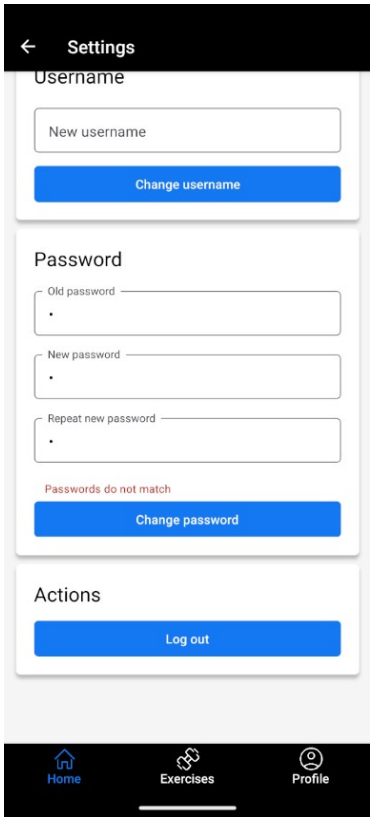
Przekroczenie licznika odpoczynku (przed)



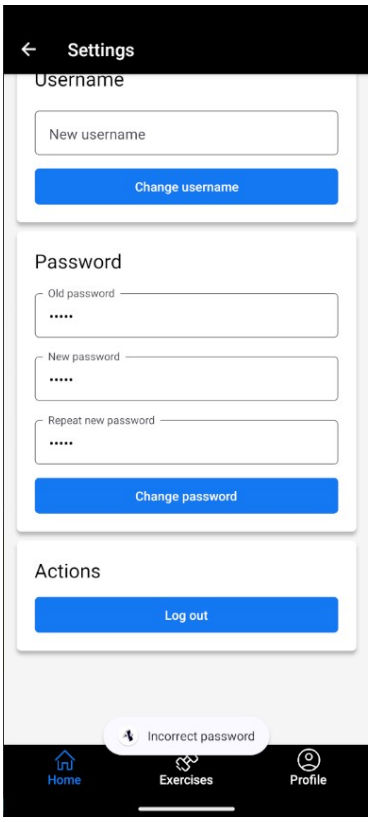
Przekroczenie licznika odpoczynku (po)



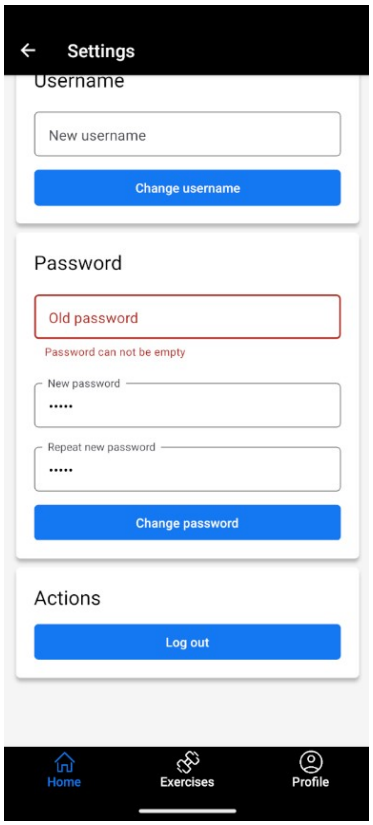
Błąd przy zmianie loginu



Niepoprawne powtórzone hasło



Niepoprawne hasło



Próba wprowadzenia pustego hasła

4. Podział Pracy

Początkowy podział pracy przedstawia się następująco:

- **Filip Stępień** - odpowiedzialny za definicję typów, ekran ćwiczeń i ekran profilu użytkownika,
- **Rafał Grot** – odpowiedzialny za ekrany logowania, rejestracji, ustawień i API,
- **Damian Karwat** – odpowiedzialny za ekran główny, ekran aktualnie wykonywanego treningu, dolny pasek nawigacji.

W dalszej części pracy zadania poniekąd nakładały się na siebie, jako że np. ekrany korzystają ze wspólnego kontekstu, bądź wchodzą w interakcje ze sobą nawzajem. Z tego powodu przydzielano zadania dynamiczne podczas rozwiązywania kolejnych problemów w aplikacji.

5. Podsumowanie

W aplikacji z powodzeniem udała się implementacja wszystkich założeń i funkcjonalności, które były planowane na początku pracy. Podczas pracy nad projektem napotkano wiele wyzwań, takich jak :

- Integracja różnych funkcjonalności w obrębie wspólnego kontekstu aplikacji,
- Zapewnienie płynnego działania na różnych urządzeniach oraz optymalizacja interfejsu użytkownika,
- Przechowywanie danych w przemyślany, ustrukturyzowany sposób, tak aby posługiwanie się nimi było proste,
- Synchronizacja danych treningowych w taki sposób aby można było elastycznie zapisywać je do API, a następnie pobierać i edytować lub usuwać,
- Implementacja komunikacji i przekazywania sygnałów pomiędzy zagnieżdżonymi komponentami aplikacji.

Wszystkie te problemy zostały pomyślnie rozwiązane. Aplikacja była testowana zarówno na emulatorach, jak i rzeczywistych urządzeniach, aby upewnić się, że działa poprawnie i zapewnia spójne doświadczenie użytkownika.

- Odciążenie kontekstu aplikacji z dużej ilości danych, aby przyśpieszyć czasy ładowania,
- Kompatybilność aplikacji z wieloma urządzeniami (pomimo poprawnego działania, zdarza się że na różnych platformach elementy GUI zachowują się inaczej).

W przyszłości aplikacja mogłaby zostać rozszerzona o dodatkowe funkcje, takie jak:

- Możliwość logowania i tworzenia konta przez np. Google lub Facebook,
- Możliwość udostępniania postępów lub zdjęć w mediach społecznościowych,
- Synchronizacja z innymi aplikacjami, np. aplikacjami do diety,
- Rozszerzone statystyki spersonalizowane dla użytkowników,
- Bardziej motywujący system śledzący progres (np. w postaci nagród lub osiągnięć).