Conclusions

This self-development application for planning and controlling the implementation of goals has met the key assumptions made at the planning stage. The user can freely plan daily activities via two views, can conveniently take notes, search data and can easily declare their daily mood. The application met both its functional and non-functional requirements (performance could probably be further improved to display very large amounts of data in one view).

The original plans also included, among others: adding a focus mode with a timer and mood statistics, which could not be implemented in the available time. However, these are good directions for the development of the application, making it a system that supports the user's self-development even more comprehensively.

The author is pleased that while creating the application he mastered the implementation of the MVVM pattern and acquired new, valuable experience in the software development process, especially in WPF technology. The goals set at the beginning motivated him not only to experiment with the possibilities of frameworks or programming languages, but also to search for publicly available knowledge and help in solutions, which increased his skills in solving technical problems. An engineer, especially an IT specialist, needs the ability to efficiently search for solutions in documentation, courses, books and Internet forums in order to not deviate from the recognized standard, benefit from the experience of wiser people and avoid common mistakes. The author had previously created WPF applications without separating them into views and view models, but now he understands not only the idea, but also the practical aspects of using MVVM. Designing and implementing applications in this way makes it easier to organize the source code and distribute responsibility for application functions within the system (which can also support testing and teamwork). The author also gained proficiency and independence in more advanced use of the XAML language.

One of the main conclusions that can be drawn from this work is that the key to creating an application is to precisely define its requirements (also in terms of its interface) before the implementation phase. This allows you to avoid situations of indecision and experimentation with the source code, thanks to which software development is carried out regularly, gradually. On the other hand, work on this application highlighted the problem of excessive perfectionism in the initial phase of work related to defining the application requirements. Such a situation can also paralyze the entire process and make decision-making difficult. Therefore, iterative, cyclical approaches to work – similar to the principles set out in the Manifesto Agile – are much more efficient. They combine work open to changes for maximizing the level of meeting user needs with minimalist work focused on a more general goal achievement, so that the software delivery time does not extend without an important reason. Work on this application also highlighted the value of a test-oriented approach already in the initial phase of software development. Although the author of the application tested it on an ongoing basis after almost every change to the source code, he could test the application using unit or integration tests along with the implementation, so as not to have to later identify and fix unexpected errors in its operation.