PinBerry: A Lightweight and Personalized System for Photo Navigation

I. Chica Becerra
Systems Engineering, Faculty of Engineering
University Distrital Francisco Jose de Caldas
ichicab@udistrital.edu.co
Bogotá, Colombia

J. S. Herrera Rodríguez
Systems Engineering, Faculty of Engineering
University Distrital Francisco Jose de Caldas
jusherrerar@udistrital.edu.co
Bogotá, Colombia

Abstract—**(i) In an increasingly visual and connected world, users need straightforward tools to view, share, and preserve images as a means of communication, inspiration, and personal expression, but many existing platforms are either too complex or overloaded with features that detract from this essential purpose. (ii) To address this gap, PinBerry is introduced as a lightweight, user-friendly application that simplifies image interaction by focusing on core functionalities: viewing, saving, and sharing photos in an intuitive and inspiring environment.**

Index Terms—**Object-Oriented Programming (OOP), Software Design, User Experience, Photo Management, Visual Content Sharing, Digital Platform, Image, Picture.**

I. Introduction

In the contemporary digital landscape, visual content has become a primary medium for communication, self-expression, and inspiration. Platforms such as Pinterest, Instagram, and Tumblr have emerged to fulfill users' needs to view, share, and organize images. However, these platforms often encompass complex features and interfaces that may overwhelm users seeking straightforward functionalities.

Pinterest has established itself as a platform for discovering and organizing visual inspiration. Its design encourages users to curate content through "pins" and thematic boards. Research indicates that engagement with Pinterest can positively impact users' emotional well-being, suggesting that inspiring content may mitigate stress and enhance daily positive emotions [1]. Instagram, focusing on photo and video sharing, offers features like filters and stories to enhance user engagement. Studies have identified various motivations for Instagram use, including surveillance, documentation, coolness, and creativity. Notably, a positive correlation exists between narcissistic traits and the frequency of Instagram use, highlighting the platform's role in self-presentation [2]. Tumblr provides a space for users to express themselves through multimedia posts, fostering communities centered around shared interests. The platform's emphasis on self-presentation significantly influences content propagation, as users tailor their posts to align with community norms and expectations [3]. These platforms employ various computational techniques to enhance the user experience. Pinterest uses algorithms to recommend content based on user behavior, aiming to personalize the discovery process. Instagram uses computer vision and clustering methods to analyze photo content and categorize user types, facilitating the delivery of targeted content [4]. In addition,social media data have been used to predict user behavior,such as purchase intentions, by analyzing profiles and activity patterns [5]. Despite the advanced features and widespread adoption of these platforms, there is still a demand for applications that offer simplicity and ease of use in managing visual content. PinBerry emerges as a solution to this need, providing a lightweight and user-friendly application that enables users to view, save, and share photos effortlessly. By focusing on core functionalities and an intuitive interface, PinBerry aims to facilitate personal expression and inspiration without the complexities associated with existing platforms.

II. Methods and materials

The development of PinBerry was structured around object-oriented programming (OOP) principles with a focus on simplicity, clarity, and long-term maintainability. Our methodology emphasizes a balance between sound engineering and practical application, aligning design decisions with real user needs and constraints. Furthermore, the system architecture and class responsibilities were guided by the SOLID principles to enhance modularity and reduce coupling.

For example, the *Single Responsibility Principle (SRP)* was respected by ensuring that each class handles only one responsibility. The *User* class is solely responsible for managing user-related data and actions, while the *Photo*, *Gif*, and *Video* classes handle multimedia-specific properties. To satisfy the *Open–Closed Principle (OCP)*, the system was designed so that new media types can be added by simply creating new classes that implement the existing *Multimedia* interface, without modifying existing code. This interface-based approach also supports the *Liskov Substitution Principle (LSP)*, as all multimedia subclasses can be used interchangeably wherever a *Multimedia* object is expected.

Additionally, the use of interfaces in place of monolithic abstract classes aligns with the *Interface Segregation Principle (ISP)*: each interface exposes only the methods relevant to its context, avoiding the imposition of unnecessary methods on implementing classes. While the *Dependency Inversion Principle (DIP)* is not applied through a full inversion of control container, the separation between the interface definitions and their concrete implementations reduces direct dependencies and prepares the codebase for future inversion mechanisms.

By adhering to both OOP fundamentals and the SOLID guidelines, we ensured that the system remains robust, testable, and easy to evolve. This section outlines the solution's conceptual design, the technical decisions that shaped its development, and the rationale behind its appropriateness.

A. *Design of the Solution*

PinBerry was conceived using the foundational principles of object-oriented design: encapsulation, abstraction, and modular responsibility. This approach allowed us to model the application in terms of discrete, self-contained objects—such as users, photos, and galleries—each responsible for specific behavior. As shown in Figure 1, the class diagram illustrates how responsibilities are distributed across well-defined classes and how they relate through inheritance and association. As Bertrand Meyer emphasizes, software quality improves significantly when objects enforce strict contracts through their interfaces, limiting external interference and ensuring predictable behavior [7].

The application architecture was intentionally kept flat and simple, avoiding deep inheritance hierarchies or overly abstract class structures. Although inheritance is a powerful OOP mechanism, we aligned with Bruce Eckel's view that it should be applied selectively and only when genuine behavioral sharing is required. Instead of generalizing prematurely, we focused on building components that solve concrete problems, deferring abstraction until the need becomes evident.

Polymorphism was selectively introduced through the implementation of the *Multimedia* interface, shared by the *Photo*, *Gif*, and *Video* classes. This interface defines common operations such as retrieving metadata (name, route, size, and date), enabling consistent behavior across media types. By relying on interface-based polymorphism, the design remains extensible while preserving clarity and cohesion. This decision aligns with recommendations from Gamma et al., who advise applying patterns and abstraction only when they deliver practical benefits in the system context.

In summary, our design strategy was not just object-oriented in structure, but also in mindset—centered on the behavior and responsibility of each object, rather than forcing generalized solutions prematurely.

B. *Technical Decisions and Trade-offs*

A core decision throughout the project was to enforce strong encapsulation, not only at the data level but also in how internal logic is exposed. As Booch highlights, encapsulation is not simply about access control; it is about reducing coupling and protecting the integrity of components [9]. In PinBerry, classes only expose methods that align with the business logic, minimizing the surface area for unintended interactions.

Although early in the project we avoided inheritance in favor of flat, self-contained modules, we later introduced a shared

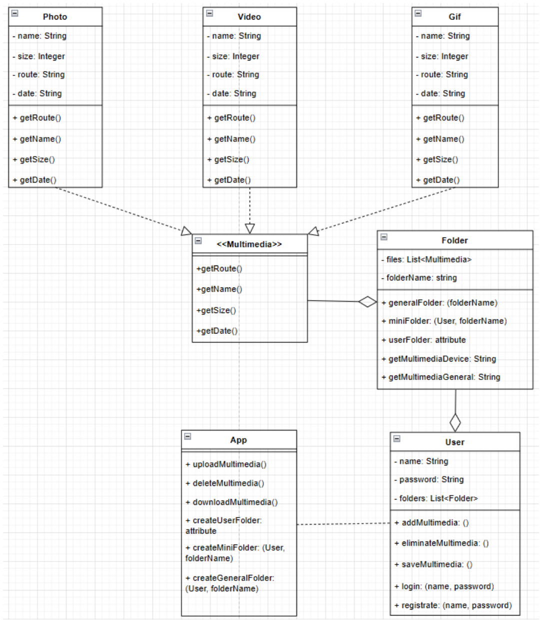

Fig. 1. PinBerry Class Diagram

interface named *Multimedia* to unify the behavior of three core content types: *Photo*, *Gif*, and *Video*. This interface defines common operations such as retrieving the name, route, size, and creation date, thus promoting consistency and reusability without compromising simplicity. By relying on interface-based inheritance, we maintain clear separation of concerns and avoid the pitfalls of deep hierarchies.

This selective use of inheritance aligns with Eckel's view that abstraction should be introduced only when a genuine need arises, and reinforces our overall design philosophy of clarity and maintainability. As Deitel notes, simplicity and transparency are critical in early-stage systems, where overengineering can obscure real functionality [11].

Our adherence to these principles enabled faster iteration and testing. Dependencies between modules were minimized, and class responsibilities were tightly scoped. These choices improved maintainability and prepared the system for future updates, such as supporting new gallery layouts or enhanced export functionalities. .

C. *Appropriateness of the Solution*

We believe that PinBerry offers an appropriate and well-structured solution for the problem at hand. Users today face increasingly complex digital ecosystems when it comes to sharing and organizing photos. Large platforms often prioritize engagement metrics over usability, resulting in cluttered experiences. PinBerry counters this by offering a streamlined, distraction-free alternative, centered on image discovery and expression.
As illustrated in the user experience diagram, the application guides users through a clear and intuitive flow - from login to deleting your own content - that ensures ease of use

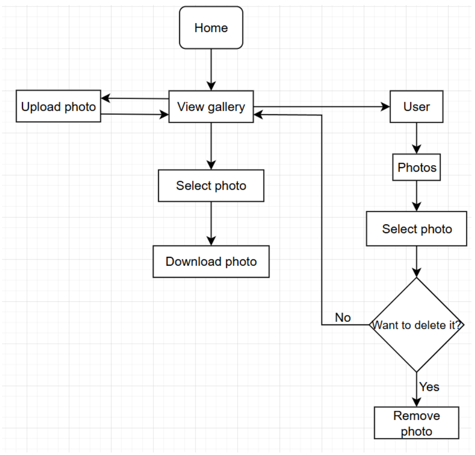

Fig. 2. PinBerry User Experience Diagram

and minimizes navigation friction. From a software design standpoint, the solution is robust yet flexible. It supports future extensibility through interfaces in strategic areas (such as storage or layout rendering), while the core domain remains grounded in stable, well-tested components. This is consistent with Gamma et al.'s recommendations on applying the Strategy and Factory patterns only where variation is likely to occur [9].
Ultimately, our approach ensures the system remains simple enough for users to engage with intuitively, while also being architecturally sound for ongoing evolution. The conscious restraint in complexity—guided by well-established OOP practices—makes PinBerry both a practical tool and a technically reliable platform.

III. Results

The current version of PinBerry successfully implements several key functionalities that align with its goal of being a lightweight and user-centric photo navigation tool. Using Java Swing, the system provides a graphical interface through which users can interact with their image collection intuitively. The implemented features include:

- **Upload Functionality:** Users can upload photos from their local storage into the application. These photos are automatically stored in both the *General Folder* (shared content) and the user's personal folder.
- **Folder Management:** Users can create and delete personalized subfolders (referred to as *MiniFolders*) to better organize their uploaded photos.
- **Gallery Viewing:** The application supports seamless navigation between the *GeneralFolder*, which aggregates all uploaded content, and the *UserFolder*, which contains only the user's personal contributions.
- **Graphical Interface:** All interactions are performed through a Java Swing GUI, providing windows, buttons, and panels that structure the flow of user interaction.

Although the authentication module (login/registration) is not yet implemented, the current prototype demonstrates functional integrity and robustness in the core workflow. The development team conducted manual testing throughout the implementation process, validating the upload, folder, and navigation features across multiple runs to ensure stability and responsiveness.

Feedback received from preliminary reviews highlighted the clarity of the technical documentation and suggested expanding GUI documentation by incorporating screenshots and a detailed explanation of the Model–View–Controller (MVC) pattern used. These improvements are currently in progress.

IV. Conclusions

PinBerry demonstrates how object-oriented programming principles can be effectively applied to build a lightweight and functional application for photo navigation. The project emphasized clarity, encapsulation, and modularity, resulting in a maintainable system that aligns with real user needs.

The application currently supports essential features such as uploading photos, organizing content into personalized folders, and navigating between shared and personal galleries—all accessible through a graphical user interface built with Java Swing. These components were developed incrementally and tested throughout the implementation process to ensure reliability and cohesion.

One key technical decision was the introduction of a shared interface, Multimedia, implemented by the Photo, Gif, and Video classes. This approach allowed for consistent behavior across media types without introducing unnecessary complexity, aligning with the project's principle of using abstraction only when it adds clear value.

Beyond the technical achievements, the development process provided meaningful learning experiences related to class design, interface structuring, and the trade-offs involved in early-stage system design. It also highlighted the importance of documentation, validation, and visual coherence in presenting a complete software solution.

Future development plans for PinBerry include refining the graphical interface to improve aesthetic appeal, preserving the original dimensions of each uploaded image (as in Pinterest's masonry layout), replacing file-based storage with a server-based system, and implementing user authentication. These enhancements aim to evolve PinBerry from a functional prototype into a more scalable and user-friendly platform.

References

[1] MAGNA, "More than a feeling: How positive platforms impact performance," *Pinterest Business*, 2023. [En línea]. Disponible en: https://business.pinterest.com/en-gb/blog/how-positive-platforms-impact-marketing-performance
[2] P. Sheldon y K. Bryant, "Instagram: Motives for its use and relationship to narcissism and contextual age," *Computers in Human Behavior*, vol. 58, pp. 89–97, 2016.
[3] J. Chang y C. Danescu-Niculescu-Mizil, "Self-Presentation Effects on Content Propagation in Tumblr," en *Proc. of the 2020 CHI Conf. on Human Factors in Computing Systems*, 2020, pp. 1–12.
[4] Y. Hu, L. Manikonda y S. Kambhampati, "What We Instagram: A First Analysis of Instagram Photo Content and User Types," en *Proc. of the Eighth Int. AAAI Conf. on Weblogs and Social Media*, 2014, pp. 595–598.

[5] Y. Zhang y M. Pennacchiotti, ”Predicting purchase behaviors from social media,” en *Proc. of the 22nd Int. Conf. on World Wide Web*, 2013, pp. 1521–1532.

[6] S. Watts, “SOLID Principles in Object-Oriented Design,” BMC Blogs, Aug. 26, 2024. [Online]. Available: https://www.bmc.com/blogs/solid-design-principles/

[7] B. Meyer, Object-Oriented Software Construction, 2nd ed. Prentice Hall, 1997.

[8] B. Eckel, Thinking in Java, 4th ed. Prentice Hall, 2006.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994

[10] G. Booch, Object-Oriented Analysis and Design with Applications, 3rd ed. Addison-Wesley, 2007.

[11] H. M. Deitel and P. J. Deitel, Java: How to Program, 8th ed. Prentice Hall, 2009.

[12] S. Oloruntoba and A. S. Walia, “SOLID: The First Five Principles of Object-Oriented Design,” DigitalOcean, Apr. 23, 2024. [Online]. Available: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

[13] J. Leskovec, A. Sharma, and A. J. Solomon, “Understanding behaviors that lead to purchasing: A case study of Pinterest,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. [Online]. Available: https://cs.stanford.edu/people/jure/pubs/pinterest-kdd16.pdf